

## **General Disclaimer**

### **One or more of the Following Statements may affect this Document**

- This document has been reproduced from the best copy furnished by the organizational source. It is being released in the interest of making available as much information as possible.
- This document may contain data, which exceeds the sheet parameters. It was furnished in this condition by the organizational source and is the best copy available.
- This document may contain tone-on-tone or color graphs, charts and/or pictures, which have been reproduced in black and white.
- This document is paginated as submitted by the original source.
- Portions of this document are not fully legible due to the historical nature of some of the material. However, it is the best reproduction available from the original submission.

# **LUMIS**

## **LAND USE MANAGEMENT AND INFORMATION SYSTEMS**

### **COORDINATE ORIENTED PROGRAM DOCUMENTATION**

(NASA-CR-149165) LUMIS: LAND USE  
MANAGEMENT AND INFORMATION SYSTEMS;  
COORDINATE ORIENTED PROGRAM DOCUMENTATION  
(Jet Propulsion Lab.) 161 p HC A08/MF A01

N77-12483

Unclas  
CSCL 08B G3/43 55821

Jet Propulsion Laboratory  
California Institute of Technology  
Pasadena, California 91103



Prepared for  
Office of Technology Utilization  
and  
Office of Applications  
NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

# TECHNICAL REPORT STANDARD TITLE PAGE

1. Report No. 43-33	2. Government Accession No.	3. Recipient's Catalog No.	
4. Title and Subtitle LUMIS LAND USE MANAGEMENT AND INFORMATION SYSTEMS COORDINATE ORIENTED PROGRAM DOCUMENTATION		5. Report Date November 1, 1976	
		6. Performing Organization Code	
7. Author(s) Nevin Bryant, Charles K. Paul, Albert J. Landini, R. Wayne Bannister, Thomas Logan		8. Performing Organization Report No.	
9. Performing Organization Name and Address JET PROPULSION LABORATORY California Institute of Technology 4800 Oak Grove Drive Pasadena, California 91103		10. Work Unit No.	
		11. Contract or Grant No. NAS 7-100	
12. Sponsoring Agency Name and Address NATIONAL AERONAUTICS AND SPACE ADMINISTRATION Washington, D.C. 20546		13. Type of Report and Period Covered Special Publication	
		14. Sponsoring Agency Code	
15. Supplementary Notes			
16. Abstract  <p>The LUMIS program has designed an integrated geographic information system to assist program managers and planning groups in metropolitan regions. Described is the series of computer software programs and procedures involved in data base construction using the census DIME file and point-in-polygon architectures.</p> <p>The system is described in two parts: (1) instructions to operators with regard to digitizing and editing procedures, and (2) application of the four data base construction algorithms: ROTATE (to achieve map registration), CHAIN (to assure the topological integrity of polygon files), DACS (DIME AREA CENTROID SYSTEM alternative to CHAIN), and PIOS (the polygon intersection overlay system for tabulating land use acreages within administrative districts).</p>			
17. Key Words (Selected by Author(s)) Earth Resources Computer Programming and Software Documentation and Information Sciences Urban Technology and Transportation		18. Distribution Statement Unclassified -- Unlimited	
19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of Pages A-IX-4	22. Price

**LUMIS**  
**LAND USE MANAGEMENT AND**  
**INFORMATION SYSTEMS**  
**COORDINATE ORIENTED**  
**PROGRAM DOCUMENTATION**

Nevin Bryant  
Project Manager

Charles K. Paul  
NASA Headquarters

Albert J. Landini  
City Planner

R. Wayne Bannister  
City Planning Associate

Thomas Logan  
Project Data Analyst

Jet Propulsion Laboratory  
California Institute of Technology  
Pasadena, California 91103

November 1, 1976

Prepared for  
**Office of Technology Utilization**  
**and**  
**Office of Applications**  
**NATIONAL AERONAUTICS AND SPACE ADMINISTRATION**



## PREFACE

This work was sponsored by the Office of Technology Utilization of the National Aeronautics and Space Administration through Contract No. NAS 7-100.

This document is one of two principal systems specifications documents to be generated by the LUMIS program. While this document addresses the problem of software and procedures involved in data base construction using the census DIME file and point-in-polygon architectures, the other<sup>1</sup> presents the problem of data base interrogation.

---

<sup>1</sup>Bryant, N.A., Yu, T.C., and Landini, A.J., LUMIS Interactive Graphics Operating Instructions and System Specifications, SP 43-31. Jet Propulsion Laboratory, Pasadena, California, August 15, 1976.

## CONTENTS

Chapter		Page
1.0	INTRODUCTION . . . . .	1-1
1.1	The System . . . . .	1-1
1.2	The Process . . . . .	1-1
2.0	DIGITIZING PROCEDURES . . . . .	2-1
2.1	Introduction . . . . .	2-1
2.2	Case 1: General Map Case . . . . .	2-1
2.2.1	Map tolerance record . . . . .	2-2
2.2.2	Control reference points . . . . .	2-4
2.2.3	Polygon description record . . . . .	2-4
2.2.4	Polygon limit record . . . . .	2-4
2.2.5	Polygon digitized boundary records . . . . .	2-6
2.3	Case 2: MMS Map Case . . . . .	2-7
2.3.1	Map tolerance record . . . . .	2-7
2.3.2	Control reference points . . . . .	2-9
2.3.3	Digitized reference points . . . . .	2-9
2.3.4	Street segment record . . . . .	2-10
3.0	CALCULATING THE STATE PLANE COORDINATES . . . . .	3-1
3.1	Introduction . . . . .	3-1
3.2	Procedure . . . . .	3-1
3.3	Calculations . . . . .	3-2
4.0	EDITING PROCEDURES . . . . .	4-1
4.1	Introduction . . . . .	4-1
4.2	General Map . . . . .	4-1
4.3	MMS Maps . . . . .	4-3
4.4	CHAIN . . . . .	4-4

PRECEDING PAGE BLANK NOT FILMED

## CONTENTS (contd)

Chapter	Page
5.0 ROTATE . . . . .	5-1
5.1 Introduction . . . . .	5-1
5.2 Methodology . . . . .	5-1
5.2.1 The scaling factor S . . . . .	5-2
5.2.2 The rotation angle $\theta$ . . . . .	5-2
5.2.3 The surveyed control points . . . . .	5-2
5.2.4 The transformation equation . . . . .	5-3
5.3 Program . . . . .	5-3
5.3.1 Phase I . . . . .	5-3
5.3.2 Phase II . . . . .	5-4
5.3.2.1 Input . . . . .	5-4
5.3.2.2 Output . . . . .	5-4
5.4 Overall Specifications . . . . .	5-4
5.5 Phase I Specifications . . . . .	5-4
5.5.1 Map tolerance record . . . . .	5-6
5.5.2 Control reference points . . . . .	5-6
5.5.3 Digitized reference points . . . . .	5-6
5.6 Phase II Specifications . . . . .	5-7
5.6.1 Polygon descriptor record . . . . .	5-7
5.6.2 Polygon limit record . . . . .	5-8
5.6.3 Polygon digitized boundary records . . . . .	5-8
5.7 Program Output . . . . .	5-9
5.7.1 The control card . . . . .	5-9
5.7.2 The minimum and maximum record . . . . .	5-9
5.7.3 The boundary record . . . . .	5-9

## CONTENTS (contd)

Chapter		Page
6.0	CHAIN .....	6-1
6.1	Introduction .....	6-1
6.2	Methodology .....	6-1
6.3	Program .....	6-4
	6.3.1 Input .....	6-4
	6.3.2 Output .....	6-4
6.4	Overall Specifications .....	6-6
6.5	Phase I Specifications .....	6-6
	6.5.1 Map tolerance record .....	6-6
	6.5.2 Control reference points .....	6-7
	6.5.3 Digitized reference points .....	6-7
6.6	Phase II Specifications .....	6-7
7.0	DIME AREA CENTROID SYSTEM (DACS) .....	7-1
7.1	Introduction .....	7-1
7.2	Methodology .....	7-1
7.3	Program .....	7-2
	7.3.1 Input .....	7-5
	7.3.2 Output .....	7-5
7.4	Phase I Specifications .....	7-6
7.5	Phase II Specifications .....	7-9
7.6	Phase III Specifications .....	7-10
8.0	POLYGON INTERSECTION OVERLAY SYSTEM (PIOS) .....	8-1
8.1	Introduction .....	8-1
8.2	Methodology .....	8-1
8.3	Program .....	8-6
8.4	Input Files .....	8-9

## CONTENTS (contd)

Chapter		Page
8.5	Output Files . . . . .	8-10
8.5.1	File 11 — major polygon file . . . . .	8-10
8.5.2	File 12 — minor polygon file . . . . .	8-12
8.5.3	Record 2 major/minor - min/max record . . .	8-13
8.5.4	Record 3 to 3+ N/4 pairs major/minor record . . . . .	8-14
8.5.5	End of file records . . . . .	8-14
8.6	Output . . . . .	8-15
8.6.1	Record 1 . . . . .	8-16
8.6.2	Record 2 thru 2+ N/4 . . . . .	8-16

## FIGURES

Figure	Page
1. LUMIS Land Use Management Information System - Organization of Coordinate Oriented Program Documentation . . . . .	1-3
2. Raw Digitized Map File Control Card Layout for Map Cases 1 and 2 . . . . .	2-3
3. Raw Digitized Map File Layout Resulting from Digitizing Map Case 1 . . . . .	2-5
4. Raw Digitized Map File Layout Resulting from Digitizing Map Case 2 . . . . .	2-8
5. Geographic Position Determination of Control Point 1 . . . . .	3-3
6. SPC Calculation for Land Use Control Point . . . . .	3-5
7. State Plane Coordinate CALCulation (SPCCAL) . . . . .	3-6
8. Deck Set-Up for ROTATE Showing Multiple Map Models . . . . .	5-5
9. Block Chaining . . . . .	6-3
10. Deck Set-Up for CHAIN for Multiple Block Boundaries . . . . .	6-5
11. Sample Block Boundary File Layout . . . . .	6-8
12. Boundary Segment Dividing Census Tract . . . . .	7-3
13. DACS Edited Polygons . . . . .	7-3
14. PIOS Stripping Method . . . . .	8-3
15. Example of Alternative Block and Land Use Intersections . . . . .	8-5
16. An Example of Identifying Unique Polygons Created by Intersecting Blocks and Land Use . . . . .	8-5
17. Reconstruction of Minor Polygons . . . . .	8-7
18. Flow Chart PIOS Program . . . . .	8-8
19. Major or Minor Polygon File Required by PIOS . . . . .	8-11

## APPENDICES

Appendix	Page
I SPCCAL Program Listing . . . . .	A-I-1
II Plotter Editing Program Listing General Map Case . . . . .	A-II-1
III Plotter Editing Program Listing MMS Map Case . . . . .	A-III-1
IV ROTATE Program Listing . . . . .	A-IV-1
V CHAIN Program Listing . . . . .	A-V-1
VI DACS Program Listing . . . . .	A-VI-1
VII PIOS Program Listing . . . . .	A-VII-1
VIII Major Polygon Data Report Program Listing . . . . .	A-VIII-1
IX Minor Polygon Data Report Program Listing . . . . .	A-IX-1



## ABSTRACT

The LUMIS program has designed an integrated geographic information system to assist program managers and planning groups in metropolitan regions. Described is the series of computer software programs and procedures involved in data base construction using the census DIME file and point-in-polygon architectures.

The system is described in two parts: (1) instructions to operators with regard to digitizing and editing procedures, and (2) application of the four data base construction algorithms: ROTATE (to achieve map registration), CHAIN (to assure the topological integrity of polygon files), DACS (DIME AREA CENTROID SYSTEM alternative to CHAIN), and PIOS (the polygon intersection overlay system for tabulating land use acreages within administrative districts).

LUMIS  
LAND USE MANAGEMENT INFORMATION SYSTEM  
Coordinate Oriented Program Documentation

1.0 INTRODUCTION

1.1 The System

The LUMIS project considers two distinct data types identified as nominal and ordinal. Nominal data refers to that information which is geographically located using a political addressing system such as the individual house and street address. Ordinal data is data geographically referenced using a classical X and Y coordinate system such as latitude and longitude. To reconcile nominal and ordinal data the LUMIS provides users with a set of Census Bureau related procedures that aggregate street addressed data to politically defined geographical zones. Those nominally labeled zones are then ordinally identified through the process of digitization with X and Y coordinates describing their perimeter.

1.2 The Process

Once the coordinate files have been produced for both the nominal and ordinal polygons the LUMIS allows for the comparison of associated data bases. Before this comparison takes place LUMIS first requires that the raw digitized coordinates be converted to state plane coordinates and that they be subjected to a graphic edit.

The graphic edit allows for the correction of a variety of syntax errors possible in the digitizing procedures. After those errors have been corrected the coordinate files are then passed to the CHAIN and ROTATE programs.

ROTATE produces a geographically true X and Y coordinate file from raw digitized or converted coordinate data referenced to any standard map system. That is done to transform coordinates oriented to a digitizing table to coordinates oriented to true ground position. In accomplishing that ROTATE considers both scale and rotation factors.

CHAIN incorporates procedures to link line segments by coordinate locations around unique political or nominally labeled polygons. Those procedures produce multiple records for polygons in which a boundary line has to be used twice, for example, once for each block found on either side of a street.

Where Census Bureau defined nominal polygons are used such as census tract or census block, it is not necessary to create a unique file of X and Y coordinates through the digitizing procedure to define their perimeters. A set of such coordinates have already been prepared by the Census Bureau through their DIME file creation efforts and those coordinates can be modified for use with LUMIS through the DACS program.

Once the nominal polygon coordinate file and ordinal polygon coordinate file have been completed, the comparison of data associated with each polygon type can take place through the map-like overlay of computer processing. This overlaying of nominal and ordinal polygons takes place in the PIOS program. The output from that effort represents the final LUMIS data product and is used as input to the LUMIS interactive graphics system, proprietary data base management systems, or the installation unique information management systems.

Figure 1 is a flow chart that illustrates the relationships between the various LUMIS coordinate oriented programs and procedures. That same figure also provides a reference to the various sections found in this Report.

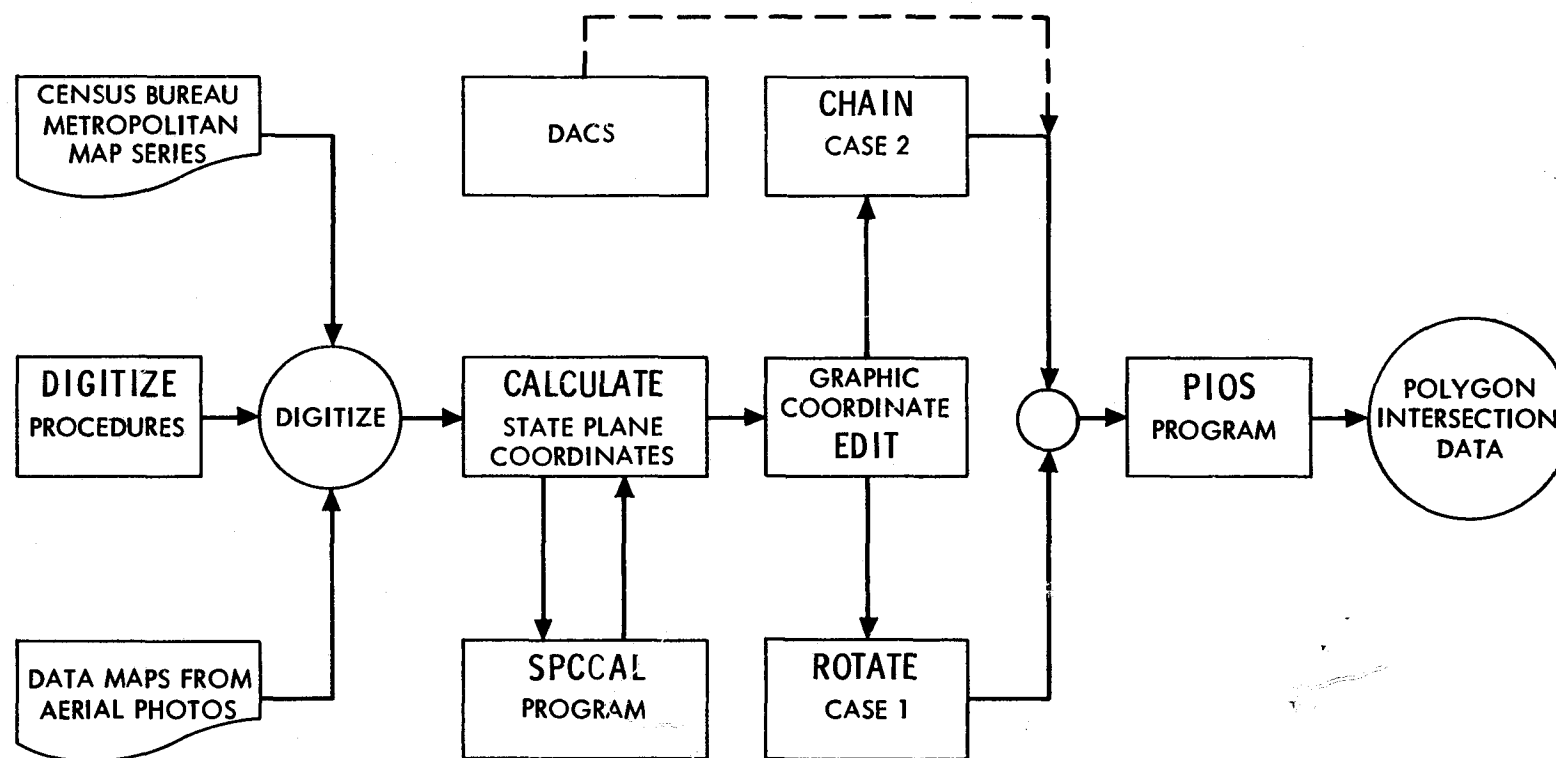


Figure 1. LUMIS Land Use Management Information System — Organization of Coordinate Oriented Program Documentation

## 2.0 DIGITIZING PROCEDURES

### 2.1 Introduction

Digitizing is the mechanical process by which X and Y coordinates are determined that ordinarily describe the outline or perimeter of any mapped polygon. In that process a piece of equipment referred to as a digitizer is used. That machine is similar in appearance and operation to a traditional drafting machine, and consists of a table and cursor.

There are numerous sets of digitizing procedures and techniques available. Their use depends on the equipment employed, the software being used, and the types of uses the coordinates are later subjected to. After reviewing these available procedures and techniques LUMIS evolved its own unique set of digitizing procedures, and they fell into two case types.

The first case type dealt with digitizing ordinal polygons and is always used with mapped and aerial photo data. The second type is for use with nominal polygons and was designed to be used only when Census Bureau DIME coordinate files were not available or nominal polygons other than those defined by the Census Bureau were to be used.

### 2.2 Case 1: General Map Case

In preparing the data maps from interpreted aerial photos the cartographer has to insure that the numbers identifying unique polygons are consecutive, i.e. (POLYNO = 1, 2, 3, 4....N). A doughnut polygon or polygon with smaller polygons within it has to be digitized with the inside polygon following the mother polygon. Thus in addition to each polygon being given a sequence number, beginning with one, it is also given a PTYPE or polygon type classification number. All mother polygons are given PTYPE numbers of 10 regardless of their POLYNO or sequence number. All mutually exclusive interior or doughnut polygons are given PTYPE numbers of 11, with inclusive interior polygons being given increasing sequential type numbers. The most interior polygon will always have the highest PTYPE number.

After completing interpretation mapping and polygon identification numbering the map is set up on the digitizer table at any orientation, and

taped down. The digitizer and keypunch are then turned on. The map setup record is established with KSET equal to 1 or other successive numbers depending on the number of map setups for a run. This is recorded on cards or tapes depending on the digitizer being used. Then the map tolerance record is created as indicated in Figure 2 and explained in the text.

2.2.1 Map tolerance record establishes the limits in calibration of the digitized coordinate reference points and the overall perimeter of the polygon to follow.

NPOLY - Total number of polygons for a map setup.

M - Number of control points provided as calculated and digitized reference records.

X0TOL, Y0TOL, and THETOL are user specified tolerance levels in the solution of a digitized origin (X0, Y0 in inches) and the rotation angle ( $\theta$  - in minutes of arc) between digitized map and accepted reference source.

Sometimes NPOLY is not known until the digitizing is finished for a given setup. Thus the NPOLY value may at times be entered after the digitization is complete. The number of control points (M) is entered for the map, followed by the tolerance levels for the digitizer origin and the rotation angle solutions (X0TOL, Y0TOL, THETOL).

The California State Plane Coordinates (SPC) records follow. Those records contain the SPC's of the M control points, and are computed at a later time according to the procedures outlined for calculating the state plane coordinates.

Control points are selected in the four corner areas as much as possible to avoid problems later in the ROTATE program's least square analysis.

The coordinates of the M control points are digitized in the same order as the SPC coordinates are to be calculated.

Control point cards (digitized reference records) are created by moving the digitizing cursor to each of the map control points, in the proper order, and then pressing the digitizer register button.

SPC DIGITIZED  
REFERENCE RECORDS

COLS.  
FORMAT  
EX.  
VAR.

COLS.	6	12	18	24	30	36	42	48	54	60	66	72
FORMAT	I6	I6	"	"	"	"	"	"	"	"	"	"
EX.	+24789	+123436	-	-	-	-	-	-	-	-	-	-
VAR.	XD(1)	YD(1)	XD(2)		YD(2)							

CALIFORNIA STATE PLANE  
COORDINATE (SPC)  
REFERENCE RECORDS

COLS.  
FORMAT  
EX.  
VAR.

COLS.			"	"	"	"	"	"	"	"	"	"
FORMAT	I9											
EX.	-	-	-	-	-	-	X(7)	Y(7)				
VAR.	-	-	-	-	-	-						

COLS.  
FORMAT  
EX.  
VAR.

COLS.	9	18	27	36	45	54	63	72	
FORMAT	I9	I9	"	"	"	"	"	"	"
EX.	44349278	42471232	-	-	-	-	-	-	-
VAR.	X(1)	Y(1)	X(2)	Y(2)					

COLS.  
FORMAT  
EX.  
VAR.

COLS.	5	10	20	30	40
FORMAT	I5	I5	F10,3	F10,3	F10,3
EX.	48	7	.040	.040	.500
VAR.	NPOLY	M	XOTOL	YOTOL	THETOL

COLS.  
FORMAT  
EX.  
VAR.

COLS.	5
FORMAT	I5
EX.	1
VAR.	KSET

MAP TOLERANCE  
RECORD

MAP SETUP RECORD

Figure 2. Raw Digitized Map File Control Card Layout for  
Map Cases 1 and 2



2.2.2 Control reference points are calculated from an existing reference source by the user. They are given in tenths of a foot without a decimal point. In this case the control points are in California Zone 7 State Plane Coordinates, but any legitimate earth coordinate system is acceptable. One through M coordinates are coded with four or less pairs per card.

Digitized reference points are entered in the same order as the control reference points in thousands of an inch, without the decimal point. One through M coordinates are coded with six or less pairs per card.

Digitization of the map polygons is then performed. (See Figure 3.) For each polygon of the map setup which corresponds to the initial setup cards of Figure 2, a polygon designation record is created using the keypunch machine directly. A polygon limit record is then created.

A polygon description record, polygon limit record, and polygon boundary record are required for each polygon digitized (NPOLY) in a map setup (KSET).

2.2.3 Polygon description record contains control information for each polygon:

MAPNO:	Map number for the digitized polygon.
MAPOL:	Total number of polygons on a map. This could, but does not necessarily equal NPOLY (number of polygons in a map setup).
LUCODE:	Identifying information code.
POLYNO:	Sequence number of polygon, 1 to MAPOL.
PTYPE:	Designates doughnut polygons, or polygons within polygons; normal polygons are digitized number 10. Numbers larger than 10 indicate succeeding doughnut polygons.

2.2.4 Polygon limit record. One limit record is required for each polygon.

XMIN, YMIN, XMAX, YMAX: The minimum X, minimum Y, maximum X, and maximum Y digitized coordinates, respectively, of the polygon.

NP: Total number of digitized coordinates in a polygon.

ORIGINAL PAGE IS  
OF POOR QUALITY

2-5

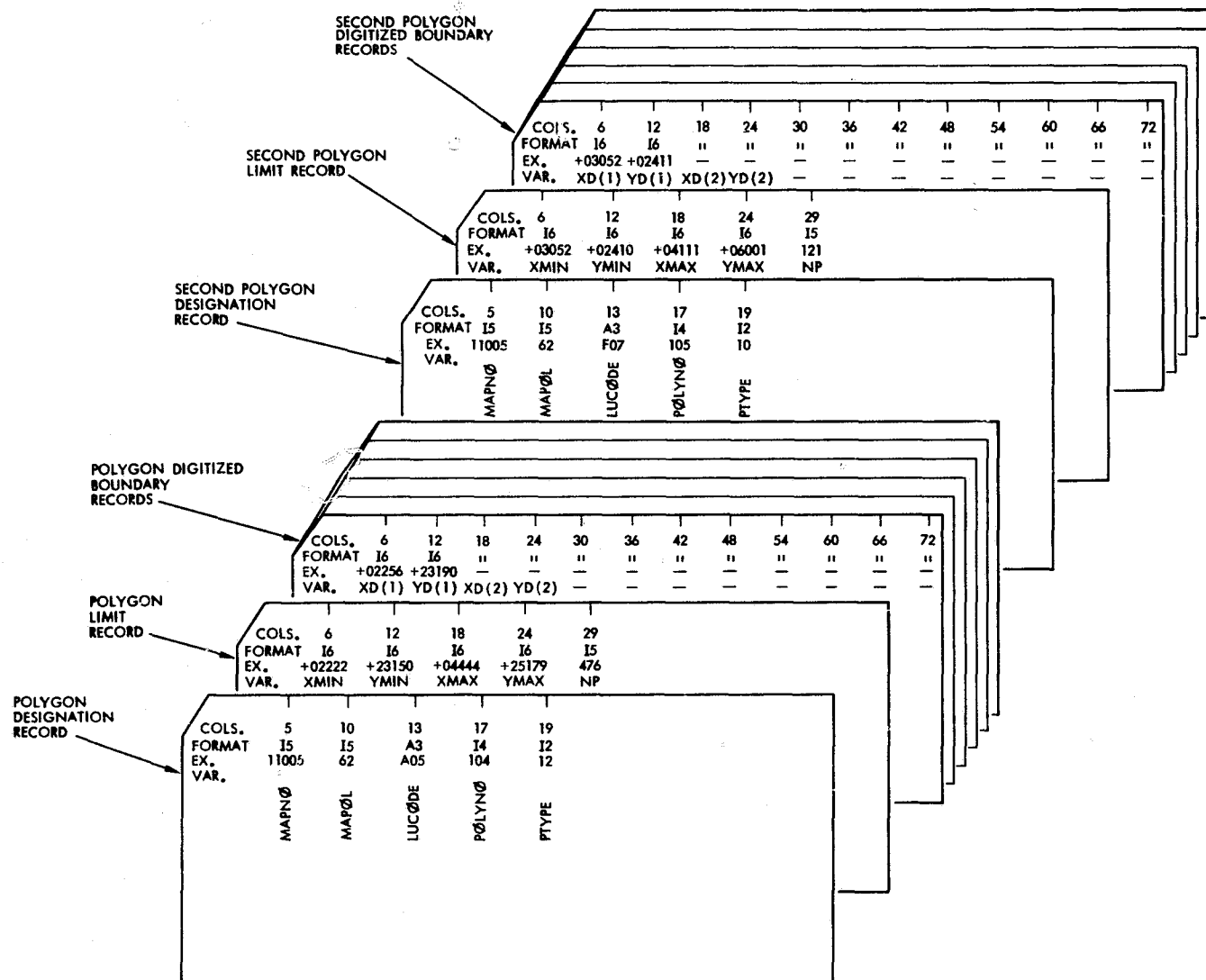


Figure 3. Raw Digitized Map File Layout Resulting from Digitizing Map Case 1

To create the polygon limit record, the digitizing cursor is set just outside of the left-most border of the polygon to be digitized. The section "left-most" implies the smallest X coordinate as defined by the digitizer table. Only this minimum X coordinate is digitized and not the Y. It must be insured that this XMIN is less than any X coordinate contained on the polygon boundary.

Next the cursor is placed on the digitizer table just below the lowest polygon point. Just the Y coordinate was digitized as YMIN.

Then the cursor is placed just to the right of the right-most polygon point and the X coordinate digitized as XMAX and finally the cursor is placed above the highest polygon point and the Y coordinate digitized as YMAX. If the digitizing cursor has no provision for entering only one component of a pair, it would be a simple matter to read the coordinate from the digital read out on the console and then enter it by hand on the terminal or keypunch.

It should be noted that the output polygon limit record contains the SPC minima and maxima, not the transformed digitized minima and maxima (if the rotation angle is large enough, the two sets of minima and maxima may be the same).

The number of points (NP) for the polygon is taken from the map (if properly prepared) or by counting the number of polygon digitized boundary records when finished. If this number is N, it is multiplied as 6 by (N-1) and this product added to the number of pairs of coordinates on the list record to obtain NP. The suggested alternative is for the photo interpreter who creates the map to indicate the digitization points at a density sufficient to adequately reflect line curvature. In so doing, he could indicate the point count before digitization commenced.

Finally, the polygon digitized boundary records are created either by running the cursor counterclockwise around the polygon boundary and digitizing at the preselected points, or at the operators' discretion. The polygon is closed by digitizing the first point again at the finish.

2.2.5 Polygon digitized boundary records. One set of digitized coordinates is required for each polygon.

XD(1), YD(1), ---XD(NP), YD(NP): Digitized coordinates of polygon.

### 2.3 Case 2: MMS Map Case

In digitizing maps of the Case 2 type users largely follow Case 1 procedures. Generally only four (4) control points are used for the MMS maps (M=4) since these maps were compiled on the State Plane Coordinate base and hence a more accurate transformation from digitizer coordinates to SPC's can be effected than with other map projections.

The control points are located and their State Plane Coordinates determined by finding the SPC tic marks on the base map's margins, and connecting the similar tics with a straight edge and hard lead (4H) pencil. Then the four intersection points are digitized and the SPC's read off the map margins. If there is an insufficient number of SPC tic marks on the map base, users should select a point in each of the four corners that can be easily identified on a USGS quad, and the SPC's identified according to the method described for calculating the State Plane Coordinates.

Digitization of the street segments is then performed, in any arbitrary order and direction. See Figure 4. All street segments are digitized for each tract, and each tract is a unit unto itself, i.e., each tract contains the header records of Figure 2 (except the map setup records, which in this case indicate the number of tracts which are to be processed). Thus each tract has a map tolerance record where NPOLY is the number of street segments in the tract. The SPC reference and SPC digitized reference records follow for each census tract, even though those records may have been precisely the same for several tracts on the same MMS map.

Blocks are digitized as units in a tract, therefore geographic reference control points must be established per each tract. Each map contains one or more tracts. A map tolerance record, control reference record and digitized reference record are also required.

2.3.1 A map tolerance record establishes the limits in calibration of the digitized coordinate reference points and the overall perimeter of the polygon to follow.

NPOLY    — Total number of blocks in a tract.

M         — Number of control points provided as calculated and digitized reference records.

ORIGINAL PAGE IS  
OF POOR QUALITY

2-8

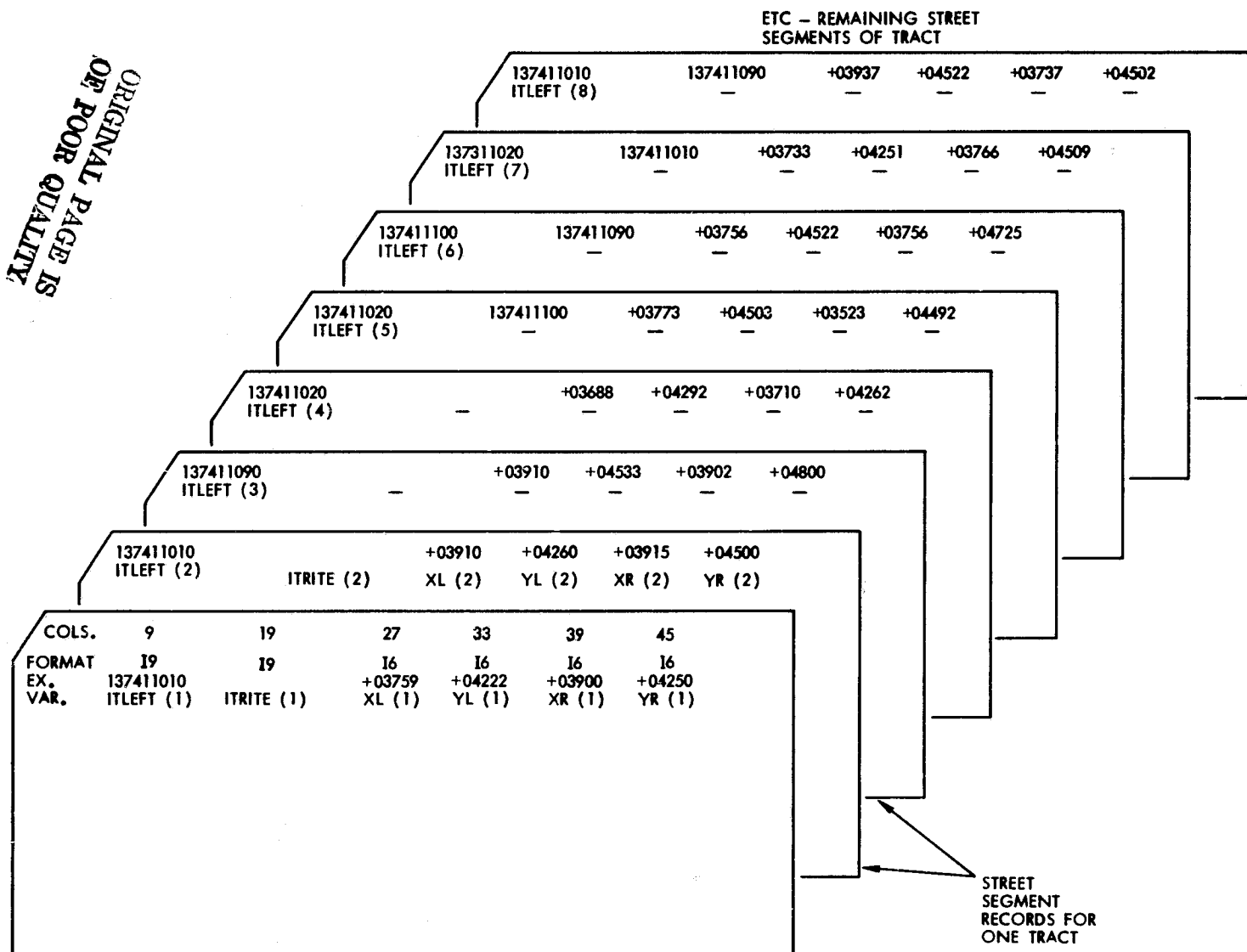


Figure 4. Raw Digitized Map File Layout Resulting from  
Digitizing Map Case 2

X0TOL, Y0TOL, and THETOL are user specified tolerance levels in the solution of digitized origin (X0, Y0 in inches) and the rotation angle ( $\alpha$  - in minutes of arc) between digitized map and accepted reference source.

2.3.2 Control reference points are calculated from an existing reference source by the user. They are given in tenths of a foot without a decimal point. In the LUMIS test case the control points were in California Zone 7 State Plane Coordinates, but any legitimate earth coordinate system is acceptable. One through M coordinates are coded with four or less pairs per card.

2.3.3 Digitized reference points are entered in the same order as the control reference points in thousands of an inch, without the decimal point. One through M coordinates are coded with six or less pairs per card.

Digitizing the MMS maps can be performed by one person as the data maps were if the census areal units are prepunched by a simple FORTRAN program. Then the digitizing operator need only to place the punched records into the keypunch feeder in the same order as the street segments which he proposed to digitize. Since the LUMIS project involved several thousand census blocks in the Santa Monica Mountains, this procedure was followed to automate and expedite the digitizing procedure. However, for a limited amount of digitizing such as that which might have been performed by the City of Los Angeles to update the census areal units each year, a two-person method was suggested. Figure 4 illustrates the card deck setup for that method.

In the "limited amount of digitizing" technique, the digitizing operator places the cursor on the node at the tail end of the arrow along the street segment. He then calls the block designator on the left of the street segment (137411010) to the person on the keypunch, who then punched this designator. Note the ".0" of the tract number (1374.01) was dropped, as redundant information, a "0" was added to the block number (101), and the tract and block numbers were joined. The implied ".0" of the block number was to allow for future block splitting and necessary decimal designators.

The tract and block number to the right of the street segment are then called out to the keypuncher who punches the number as shown. The keypuncher then presses the space bar. The digitizing operator presses the

digitizing button, thus recording on the card the X, Y coordinates of the starting node. The digitizing operator moves the cursor along the arrow to the ending node of the segment and again presses the button thus recording the digitizing table coordinates of this node. The keypuncher releases this card and sets a new card into the keypunch for the next street segment.

The digitizing operator meanwhile marks this segment on the map so that he will not redigitize it later. The procedure then is repeated, segment by segment, in any arbitrary order, until the tract is finished. Note that the tract right field for segments on the census tract boundary (that is, the designator for an area outside of the tract presently being digitized) is simply left blank. In the case where a census block is entirely within another census block (forming what is termed a "Doughnut Hole"), the block right is also left blank.

Link records for each street segment in a census tract are required.

2.3.4 Street Segment Record. Each record contains the left and right census tract/block for a segment and the beginning and ending coordinates. If a segment is missing a tract/block right, the coordinates are coded but the right position is left blank.

ITLEFT = Tract/block left of street segment (1 to MPOLY)

ITRITE = Tract/block right of street segment (1 to NPOLY)

XL, YL = X and Y coordinates of the beginning made for segment.

XR, YR = X and Y coordinate of the ending node for a segment.



### 3.0 CALCULATING THE STATE PLANE COORDINATES

#### 3.1 Introduction

The following discussion describes the theory and methodology for calculating the State Plane Coordinates (SPCs) of the control points used in the digitizing process. These procedures are only used for the General Map Cases because the SPCs for the special MMS Map Case are determined at the time of digitization. The tools required for calculating the SPCs include:

1. New, unfolded USGS Quad
2. Digitizer
3. SPCCAL computer program (Appendix I)

#### 3.2 Procedure

Place the new USGS quadrangle on the digitizer table, and align it parallel to the bottom and sides. Use the digitizer to insure that the X coordinate of the left map edge (not paper edge) differs by no more than .001 inch between the top and bottom of the map. (If the paper has stretched, this accuracy cannot be obtained; secure a new quad.) Use the same method for the Y coordinate of the bottom left and right map edges. Then tape the quad securely and mark the control points on the map. Locate the four 2-1/2 minute and longitude tic marks surrounding each of the control points.

It is suggested that these 2-1/2 minute tic marks be used for determining the control points' SPCs instead of the actual SPC tic marks provided on the map margins for two reasons: (1) there is an insufficient number of tic marks on the margin to establish a dependable grid on the topographic map, especially when considering that (2) the SPC grid cannot be linear on a topographic map projection.

The State Plane Coordinates for the 2-1/2 minute intersection tics can then be obtained from the U. S. Coast and Geodetic Survey Plane Coordinate Intersection Tables (2-1/2-minute), SP-327, a copy of which can be obtained from the Los Angeles County Engineer's Office or the Superintendent of Public Documents.

With the SPCs of the four bounding 2-1/2 minute intersections known, they can be designated  $(X_1, Y_1)$ ,  $(X_2, Y_2)$ ,  $(X_3, Y_3)$ , and  $(X_4, Y_4)$  representing the southwest, southeast, northwest, and northeast tic marks respectively. (See Figure 4.) Since the SPCs form a curvilinear coordinate system on the topographic map, a four-way interpretation for the X and Y coordinates of each control point can then be made.

Using the digitizer, establish the origin at the first control point and measure (to the nearest .001 inch) the X and Y displacements to two diagonal 2-1/2 minute intersections. The four resulting numbers are named  $x_1, x_2, y_1, y_2$ , and refer to the segments created by an imaginary perpendicular north-south line drawn through the center of the control point (Figure 5). To obtain the most accurate results, it is best to remeasure the four distances several times (reoriginating on each occasion), then average the results. Repeat this procedure for each control point, proceeding in the same order as the points were digitized on the mylar base map.

### 3.3 Calculations

The four SPC solutions for each of the X and Y coordinates of the control points can now be calculated by:

$$X = X_1 + 2000x_1 + \frac{y_1}{y_1 + y_2} (X_3 - X_1)$$

$$X = X_3 + 2000x_1 - \frac{y_2}{y_1 + y_2} (X_3 - X_1)$$

$$X = X_2 - 2000x_2 + \frac{y_1}{y_1 + y_2} (X_4 - X_2)$$

$$X = X_4 - 2000x_2 - \frac{y_2}{y_1 + y_2} (X_4 - X_2)$$

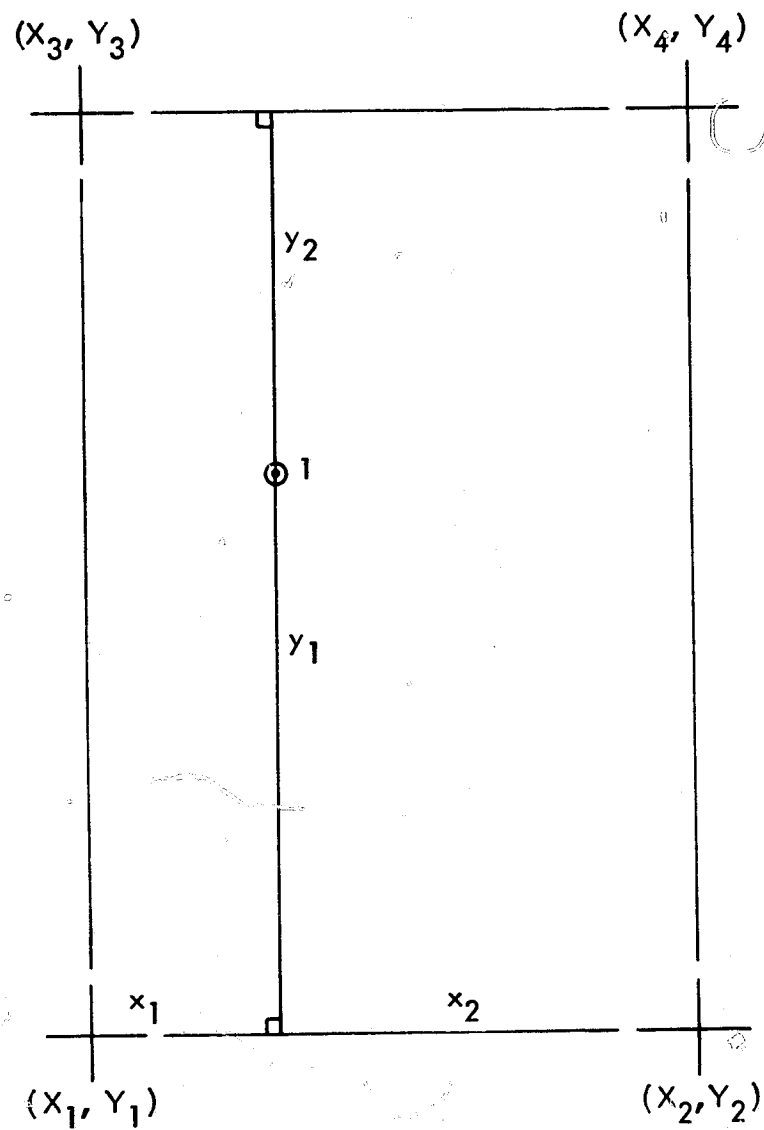


Figure 5. Geographic Position Determination of Control Point 1

$$Y = Y_1 + 2000y_1 - \frac{x_1}{x_1 + x_2} (Y_1 - Y_2)$$

$$Y = Y_2 + 2000y_1 + \frac{x_2}{x_1 + x_2} (Y_1 - Y_2)$$

$$Y = Y_3 - 2000y_2 - \frac{x_1}{x_1 + x_2} (Y_3 - Y_4)$$

$$Y = Y_4 - 2000y_2 + \frac{x_2}{x_1 + x_2} (Y_3 - Y_4)$$

The SPCCAL program performs the above mathematics. The four X and Y solutions should always agree to within 3 to 30 feet of each other. If the corresponding range of solutions should ever exceed 30 feet, the  $x_1$ ,  $y_1$ ,  $x_2$ ,  $y_2$  distances should be remeasured and run through the SPCCAL program until they meet the specified tolerance. The SPCCAL program prints an error message when the tolerance is exceeded. An example of the calculations for the SPCs of a land use control point is exhibited in Figure 6.

The State Plane Coordinates of the control points may then be punched onto cards for inclusion in the ROTATE program (see Figure 7).

## LAND USE MAP 17001

ELEVATION: 900 FEET

## GRID TICS

	<u>SW1</u>			<u>SE 2</u>			<u>NW 3</u>		<u>NE 4</u>				
LATITUDE	34°	07'	30"	34°	07'	30"	34°	10'	34°	10'			
LONGITUDE	118°	40'		118°	37'	30"	118°	40'	118°	37'	30"		
X	4085806.8			4098417.5			4085856.4		4098460.8				
Y	4158058.9			4158020.2			4173221.8		4173183.2				
$x_1$	=	4.455 INCHES		$x_3$	-	$x_1$	=	49	$y_1$	-	$y_2$	=	39
$x_2$	=	1.840 INCHES		$x_4$	-	$x_2$	=	43	$y_3$	-	$y_4$	=	39
$y_1$	=	6.610 INCHES		$y_1 / \Sigma y$	=	.8732		$x_1 / \Sigma x$	=	.7077			
$y_2$	=	0.960 INCHES		$y_2 / \Sigma y$	=	.1268		$x_2 / \Sigma x$	=	.2923			

	<u>X</u>	<u>Y</u>
	4094760	4171251
	4094760	4171251
	4094776	4171274
	4094776	4171274
AVERAGE:	<hr/> 4094768	<hr/> 4171262

3-5

SPCCAL IS A COMPUTER PROGRAM THAT CALCULATES THE CONTROL POINT  
SPCs FOR INPUT TO THE ROTATE PROGRAM.

DECK SETUP:

	<u>COLUMN</u>	<u>VARIABLE</u>	<u>DESCRIPTION</u>
CARD 1			
	1-5	NUMAP	NUMBER OF MAPS FOR WHICH CONTROL POINT SPCs ARE TO BE CALCULATED IN THIS COMPUTER RUN. (INTEGER)
CARD 2			
	1-10	MAPNO	MAP NUMBER (EX. 17001) FOR WHICH SPCs ARE TO BE CALCULATED. (INTEGER)
	11-15	NCONPT	NUMBER OF CONTROL POINTS. (INTEGER)
CARD 3			
	SPC COORDINATES OF THE FOUR 2-1/2 MINUTE LATITUDE-LONGITUDE TIC MARKS SURROUNDING THE FIRST CONTROL POINT. (SEE FIGURE (4).) PUNCH THESE COORDINATES (TO ONE DIGIT AFTER THE DECIMAL POINT) IN THE FORM OF $x_1, x_2, x_3, x_4, y_1, y_2, y_3, y_4$ , RIGHT JUSTIFIED TO EVERY TENTH COLUMN.		
CARD 4			
	$x_1, x_2, y_3, y_4$ (IN THAT ORDER) DIGITIZER MEASURED DISTANCES (IN INCHES) TO THE 2-1/2 MINUTE INTERSECTIONS.		
REPEAT ONE SET OF CARDS 3 AND 4 FOR EACH CONTROL POINT.			
FOR THE SECOND MAP, REPEAT CARD 2 FOLLOWED BY THE NECESSARY SETS OF CARDS 3 AND 4.			

Figure 7. State Plane Coordinate CALCulation (SPCCAL)

## 4.0 EDITING PROCEDURES

### 4.1 Introduction

Editing the MMS Census Tract and General Map Raw Digitized Map Files is the process through which all syntax errors are removed for input to the CHAIN or ROTATE programs. The procedures for editing the two different map cases are fundamentally the same, but the individual steps are performed with varying emphasis.

The following items are needed in the editing process:

1. Original mylar base map
2. Plotter program and plot of digitized data
3. Printout of raw data
4. Light table

The mylar base is the actual map that was digitized. A listing of the plotter editing program for the general map case is in Appendix II, and for the special MMS map case Appendix III. The editing programs were written for a UNIVAC 1108 Computer using CALCOMP plotter commands.

### 4.2 General Map

Taking the mylar base map, overlay it on top of each individual plot, and search for obvious errors like missing polygons, truncated or incorrectly digitized boundaries, extraneous lines, off registration, and control point errors. Small gaps and overlaps between polygon boundaries are normal and should not cause concern. Minor boundary errors are best repaired by locating the coordinates of the neighboring polygon's common boundary, and substituting those coordinates in the problem polygon. To use this method, plot the problem border on graph paper marking the location of each coordinate. Note the beginning and end point of the error segment, and locate similar points in the neighboring polygon.

Insert the correct coordinates and rectify the polygon's total number of points (NP) if there was a change.



This method is rather tedious, however, so if the boundary error is serious, it is easier to throw out the polygon and redigitize it as a separate setup within the map.

Often times the raw data cards become shuffled or blank cards manage to mix with the data. These types of errors show up as extraneous lines criss-crossing inside or through polygons. Off-registration is usually caused by the plotter, digitizer, or digitizer person losing origin. In most cases, the error can be corrected by determining the X and Y displacement, pulling the polygon(s) out, and treating them as a separate setup with their control points being a combination of the original control points and the displacement.

If the general map was digitized in several setups rather than all at once, inspect the plot of each setup individually, then combine them (using the plotted control points) over a light table and check for off-registration, bad control points, and missing polygons.

There are a number of errors that the visual inspection of the plotted maps will not find. Among these include incorrect land use codes (LUCODE), number of points (NP), polygon types (PTYPE), and minimum-maximum coordinates. The plotter editing program printout is most useful in spotting these kinds of problems. Its main feature (other than to create the plot) is that it counts the number of points in each polygon, and prints that number out. The most common error by far, is an incorrect NP, and such an error will always abort the ROTATE program. The editing program also prints out the regular polygon information like MAPNO, LUCODE, POLYNO, NP, digitized data, etc., but in an easily read format.

The procedure for editing the NP is simply to scan the printout comparing the computer counted NP against the digitizer's specified NP. This method not only catches wrong NPs, but also out-of-sequence header cards ("polygon designation records") and min-max cards ("polygon limit record"), and also blank cards. As for checking the LUCODE and PTYPE, each polygon in the printout must be individually checked against the mylar base map.

It is very important that no PTYPE errors pass uncorrected, as one error could falsify the polygon overlay area calculations of many neighboring polygons. For the same reason, if a polygon is removed for redigitizing as a separate setup (and all redigitized polygons must become separate setups),

careful note must be taken to determine if it is a polygon PTYPE 11 or greater. If it is, then all the polygons from the previous PTYPE 10 to the last PTYPE 11 or greater in the series (i.e., all polygons within the PTYPE 10) must also be redigitized in the new setup. This is because the polygon overlay program subtracts the area of all PTYPE 11 (or greater) polygons from the polygon they are within. Users should replot and reedit map setups in which there were a lot of errors. Redigitized polygons also should be replotted. Once all this editing is done, the raw digitized map file is ready to be inputted to the ROTATE program.

#### 4.3 MMS Maps

The MMS census tract plotter editing program (Appendix III) produces two plots of each census tract. Along each line segment in the first plot, the sequence number describing the order in which the segment was digitized is printed. In the second plot, the block number of the polygon on either side of the segment is printed. The procedure for editing the MMS raw digitized map file involves the visual comparison of the plotted census tract shapes with the actual base map.

The base map can be overlaid directly on top of the plot, but this method does not work well when the street (line) segments stimulate curved block boundaries as in hilly areas. The kinds of errors encountered include incorrect block numbers, double digitized street segments, missing segments, common nodes greater than tolerance (.045 inch) apart, and incorrectly digitized "doughnut holes". (A doughnut hole census block should be treated like a census tract, i.e., the outside block number left blank.) Between census tracts, checks must be made to insure that the common boundaries are similar. Adding new nodes is accomplished by drawing an X and Y axis through a close existing node, measuring the X and Y displacement to the new node, and adding that displacement to the coordinates of the known node. (An existing node's coordinates are located by referring to the sequence number of its line segment, then finding that line in the editing program printout.) When the entire census tract has been corrected, it should be replotted and reedited again.

#### 4.4 CHAIN

The next step is to run the raw digitized data through the CHAIN program, which will flag all the remaining uncaught errors and print the street segments in the bad census block. The error can be found by closely reexamining the census block's plot, or by manually CHAINing the street segment coordinates in the printout until two nodes are found to be greater than tolerance distance (.045 inch) apart. A common error is to have an incorrectly numbered street segment not only abort the census block it is missing from, but also the census block of the wrongly specified number. When using the CHAIN program for editing purposes, it is a good idea to modify the program so that it does not produce punched cards at each turn-around. When all the blocks finally CHAIN properly, the program can then be run conventionally to convert the digitized street segments to State Plane Coordinates.

## 5.0 ROTATE

### 5.1 Introduction

The ROTATE program is written in FORTRAN IV and produces a geographically true X and Y coordinate data base from raw digitized data referenced to any standard map system. These ROTATE corrected coordinates then act as input to PIOS (Polygon Information Overlay System) where polygon intersection statistics are calculated. Raw digitized coordinates such as those corrected by ROTATE are produced when geographically oriented researchers decide to incorporate thematically mapped data, like soils data, geological data or vegetative data into a LUMIS data base. These data maps when digitized are not usually assigned true X and Y coordinates, but rather coordinates that are relative to the digitizer table and the position of the data map to that table. Hence ROTATE's purpose is to adjust those raw digitized coordinates to their relative true ground locations.

### 5.2 Methodology

To produce geographically true X and Y coordinates the ROTATE program is broken into two major components. The first component operates with the data cards prepared for each unique data map. These initial map setup records describe the geographic coordinates of the digitizer table origin, and the table-to-geographic coordinate system relationship from which a scaling factor and rotation angle are computed. In the first component of ROTATE there is a check that insures control point accuracies meet user specified tolerances.

That check for control point accuracies requires a digitizer table-to-ground coordinate transformation. This transformation is solved by program ROTATE using a least squares solution having four unknown variables. These unknowns are:

- 1) XO . . . . . Control map X axis ground location
- 2) YO . . . . . Control map Y axis ground location
- 3) SCos  $\theta$  . . . . . Scaling Factor S times the cosine  $\theta$
- 4) SSin  $\theta$  . . . . . Scaling Factor S times the sine  $\theta$

5.2.1 The scaling factor S is expressed in a ground coordinate system of feet per inch on the map.

5.2.2 The rotation angle  $\theta$  representing the difference between the map orientation on the digitizing table and true North-South is expressed in degrees, minutes and seconds. The scaling factor and rotation angle are calculated from  $S \cos \theta$  and  $S \sin \theta$  in a simple trigonometric equation.

These variables are then used in the least square transformation equation R as follows:

$X = X + SNx$	Transformation Equation
$S =$	Scaling Factor
$X = (XO)$ $(YO)$	Rotated Coordinates
$N = (\cos \theta \sin \theta$ $-\sin \theta \cos \theta)$	Rotation Angle
$x = (XD_i), i = 1, M$ $(YD_i)$	Digitized Coordinates
$X = (X_i), i = 1, M$ $(Y_i)$	Surveyed SPC

ROTATE uses the least square approach to increase the accuracy of the basic transformation equation. This approach will produce a transformation factor (SN) that when applied will change all digitized coordinates to their ground truth equivalents. In the transformation equation, surveyed X-Y coordinates (X) and digitized X-Y coordinates for the control points (x) are expressed as arrays, M sets in length.

5.2.3 The surveyed control points ( $X_i, Y_i, i = 1, M$ ) correspond to the digitized control points ( $XD_i, YD_i, i = 1, M$ ). In the transformation equation the error between surveyed and digitized control points is averaged using the least squares method.

5.2.4 The transformation equation used in ROTATE creates two equations per control point, one for X and one for Y, so only two control points are required,  $M = 2$ . However, the least squares solution can accommodate up to eight points. For greater accuracy the maximum number are suggested.

Using the computed transformation factor, each digitized control point X and Y coordinate is entered into the equation and the measured coordinate is solved for. If this computed equivalent of the surveyed X-Y coordinate is not found to be within the used specified tolerances (X0TOL, Y0TOL, or THETOL), a message is printed out by the program indicating the degree of error and processing is stopped.

For example, a tolerance of 0.045 was used in the LUMIS Santa Monica Mountains Study. This was felt appropriate because LUMIS is a block level system and the boundaries should rarely be in error of more than the width of a secondary street, 84 ft. At a scale of approximately 1 inch = 2,000 ft., an error of 0.045 inches on the digitizing table corresponds to a geographical ground error of 90 ft.

Once the equation is formulated and tested the remaining polygons are rotated through angle  $\theta$  and scaled by the factor S. Each map set up must follow this same procedural construction of an equation, testing and processing of the polygons which follow.

### 5.3 Program

ROTATE written in FORTRAN IV, consists of a main program having two phases, and two subroutines. The subroutines perform matrix inversion and multiplication required by the least square algorithm. The main program itself is divided into two logical phases.

5.3.1 Phase I. The main program first phase calculates the unknown in a least squares equation for each map set-up from the calculated and the digitized control coordinates. The equation is tested by reentering the digitized control points to determine if the calculated control points can be found within user specified tolerances.

5.3.2 Phase II. The second phase reads data points for each polygon, enters them into the equation formed in Phase I. It then makes the adjustment in the digitized coordinates for rotation angle and scaling factor, and writes the adjusted coordinates to an output file.

There are two subroutines used in ROTATE, RATMUL and INVERSE. Subroutine RATMUL multiplies a matrix times its transpose. The regression algorithm performs this task on the original digitized matrix and on its inverse as part of the least squares formula. Subroutine INVERSE defines the inverted matrix of the original digitized data and its transpose.

5.3.2.1 Input. The ROTATE program uses as its input a card file with the following sets of information; the control specifications, the digitized calculated control points, and the digitized polygon min-max and boundary points, Figure 8.

5.3.2.2 Output. The ROTATE program produces as output a listing of transformation solution, rotation angle and a scaling factor, input digitized coordinates for each polygon, a listing and deck of transformed geographic coordinates, and any error condition messages that are invoked.

#### 5.4 Overall Specifications

The polygons are entered as sets (KSET) within an established geographic reference system, or map setup. One map may be coded as one or more setups depending on the number of times a reference is required. The first card in the deck is coded as follows:

COL        5

4  
KSET

#### 5.5 Phase I Specifications

Phase I requires as input the following information: map tolerance record, control reference record, digitized reference record.

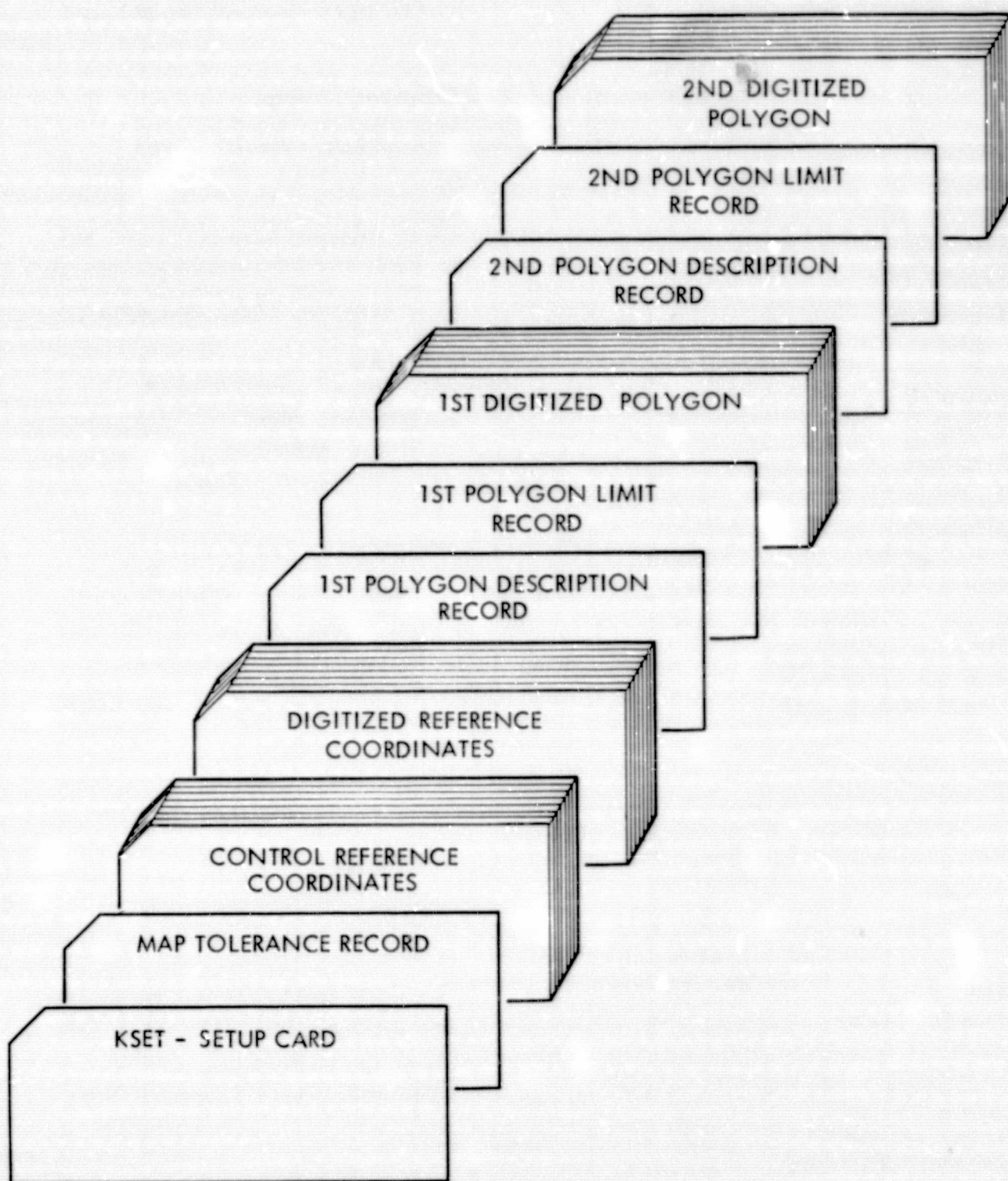


Figure 8. Deck Setup for ROTATE Showing Multiple Map Models



5.5.1 Map tolerance record establishes the limits in calibration of the digitized coordinate reference points and the overall parameter of the polygon to follow.

NPOLY - Total number of polygons for a map setup.

M - Number of control points provided as calculated and digitized reference records.

X0TOL, Y0TOL, THETOL, user specified tolerance levels in the solution of digitized origin (X0, Y0 in inches) and the rotation angle ( $\theta$  - in minutes of arc) between digitized map and accepted reference source.

COL	5	10	20	30	40
	48	7	.045	.045	.500
	NPOLY	M	X0TOL	Y0TOL	THETOL

5.5.2 Control reference points are calculated from an existing reference source by the user. They are given in tenths of a foot without a decimal point. In this case the control points are in California Zone 7 State Plane Coordinates, but any legitimate earth coordinate system is acceptable. One through M coordinates are coded with four or less pairs per card in the following format:

COL	9	18	27	36	ETC...	72
	44349278	42471232				
	X(1)	Y(1)	X(2)	Y(2)	ETC....	

5.5.3 Digitized reference points are entered in the same order as the control reference points in thousands of an inch, without the decimal point. One through M coordinates are coded with six or less pairs per card in the following format:

COL	6	12	18	24	30	36	ETC....	72
	+24789	+123436						
	XD(1)	YD(1)	XD(2)	YD(2)			ETC....	

Upon completion of Phase I the user will receive a message as to the status of the job. The message indicates the calculated scaling factor and rotation angle. If the transform equation fails the tolerance checks, an error message appears.

Problems that arise in rotating data points include syntax errors and/or incorrectly digitized or poorly placed control points. In either case, it is suggested that the program's punched output be deleted until a normal termination is achieved in the first phase.

There are two possible means to correct the error condition without redigitizing the control points: either reduced the number of data points or increase the acceptable tolerances. Since a minimum of two points are required, it is possible to discard all but the last two data points. In the cases where variation is slight, the acceptable tolerance can be increased.

In extreme cases of inaccuracy or when the data points are poorly placed, i. e., too close together or in a linear arrangement, they should be recoded. Recoded control points should make use of the old points, adding only those control points necessary to correct the problems of improper placement.

## 5.6 Phase II Specifications

If Phase I is completed without errors, Phase II begins with a polygon description record, polygon limit record, and polygon boundary record. One set of these records are required for each polygon digitized (NPOLY) in a map setup (KSET).

### 5.6.1 Polygon descriptor record contains control information for each polygon:

MAPNO: Map number for the digitized polygon.

MAPOL: Total number of polygons on a map. This could, but does not necessarily equal NPOLY (number of polygons in a map setup).

LUCODE: Identifying information code.

POLYNO: Sequence number for polygon, 1 to MAPOL

PTYPE: Designates doughnut polygons, or polygon without polygons;  
normal polygons are digitized number 10. Numbers  
larger than 10 indicate succeeding doughnut polygons,

COL	5	10	13	17	19
	11005	62	A05	104	12
	MAPNO	MAPOL	LUCODE	POLYNO	PTYPE

5.6.2 Polygon limit record. One limit record is required for each polygon. It should be noted that the output polygon limit record contains the SPC minima and maxima, not the transformed digitized minima and maxima (if the rotation angle is large enough, the two sets of minima and maxima may be the same). These limits (in SPC's) are important since they "box in" the map polygons for efficiency in the PIOS overlay techniques.

XMIN, YMIN, XMAX, YMAX: The minimum X, minimum Y, maximum X, and maximum Y digitized coordinates respectively of the polygon.

NP: Total number of digitized coordinates in polygon. Format follows:

COL	6	12	18	24	27
	+02222	+23150	+04444	25179	476
	XMIN	YMIN	XMAX	YMAX	NP

5.6.3 Polygon digitized boundary records. One set of digitized coordinates is required for each polygon.

XD(1), YD(1), ---XD(NP), YD(NP): Digitized coordinates of polygon.

Format follows:

COL	6	12	18	24	30	36	ETC....	72
	+02256	+23190						
	XD(1)	YD(1)	XD(2)	YD(2)				

## 5.7 Program Output

The program produces a card file which can later be used directly in PIOS as the minor polygon file. The cards are grouped by polygon with a control card, minimum-maximum card and polygon record description.

### 5.7.1 The control card contains the following data formatted as indicated:

Column	Format	Descriptor
1-5	I5	Polygon Descriptor (MAPNO)
6-10	I5	Number of Polygon on a Map (MAPOL)
11-13	A3	Land Use Code Identification (LUCODE)
14-17	I4	Sequence Number of Polygon (POLYNO)
18-19	I2	Polygon Type, complete or Doughnut Polygon (PTYPE)

### 5.7.2 The minimum and maximum record is formatted as follows:

Column	Format	Descriptor
1-9	I9	Minimum X coordinate value of the Polygon (XMIN)
10-18	I9	Minimum Y coordinate value of the Polygon (YMIN)
19-27	I9	Maximum X coordinate value of the Polygon (XMAX)
28-36	I9	Maximum Y coordinate value of the Polygon (YMAX)
37-41	I5	The number of coordinate pairs describing a Polygon (N)

5.7.3 The boundary record is recorded as X and Y values in a 8I9 format, four pairs per card. The number of cards to follow is determined by the data points N divided by four rounded up.

## 6.0 CHAIN

### 6.1 Introduction

CHAIN was created to prepare a coordinate data base, referenced to a standard geographic coordinate, as input to PIOS (Polygon Information Overlay System) from a raw digitized street network and political boundary such as a metropolitan map series.

### 6.2 Methodology

CHAIN also makes use of the basic least squares transformation equations with four unknowns developed for ROTATE. The table-to-ground equations use the same unknowns as those in ROTATE;

- 1)  $X0$  . . . . . Control map X axis ground location
- 2)  $Y0$  . . . . . Control map Y axis ground location
- 3)  $SCos\theta$  . . . . . Scaling Factor S times the cosine  $\theta$
- 4)  $SSin\theta$  . . . . . Scaling Factor S times the sine  $\theta$

The scaling factor S is expressed in a ground coordinate system of feet per inch on the map.

Two to four control points are required by the CHAIN Program. Since the original MMS maps are more accurate than ordinal input maps used by ROTATE, four coordinates are sufficient to maintain the same level of accuracy as eight in ROTATE. The least squares solution is again tested against user specified tolerances.

CHAIN also incorporates an algorithm to link segments by node coordinate around digitized blocks and to produce multiple records for those units which a boundary segment must use twice.

CHAIN must link digitized street segment nodes and check the polygon for completeness before segments can be transformed. Thus CHAIN will correct for minor digitizing errors or point to the node where major errors exist.

The technique used to build the block file starts by digitizing records for each link of the block shown in the top half of Figure 9.

Looking at Blocks 101, 102, 109, and 110, the coordinates are digitized in the following manner:

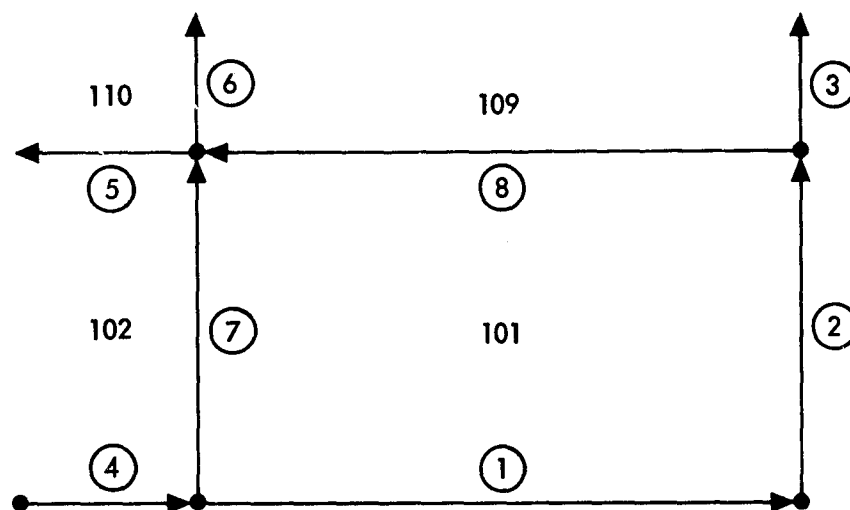
	<u>Block Lft</u>	<u>Block Rt</u>	<u>Begin</u>	<u>X-Y</u>	<u>FHD</u>	<u>X-Y</u>
1)	101		+03759	+04222	+03900	+04250
2)	101		+03910	+04260	+03915	+04500
3)	109		+03910	+04533	+03902	+04800
4)	102		+03688	+04292	+03710	+04262
5)	102	110	+03773	+04503	+08523	+04472
6)	110	109	+03756	+04522	+03756	+04725
7)	102	101	+03733	+04251	+03766	+04509
8)	101	109	+03937	+04522	+03777	+04502

In the Chaining Program each record which has two sides, (5) (6) (7) and (8) are "reversed - duplicated," that is the left block becomes the right block, from and to coordinates are reversed, and the segment is duplicated.

Segments not under consideration are then "erased" from the duplicated file. These records, tract boundary records, will be considered at a later time.

The file is sorted by the left block producing the file shown in the bottom half of Figure 9. The coordinates are then "matched" by ending and beginning nodes for each block according to user specified mapping tolerance (X0TOL, Y0TOL). Cul-de-sacs fall at the end of each block segments. They are retained only for graphic presentation and do not provide any information to PIOS.

The coordinates accepted by the CHAIN edit program are averaged, and transformed to state plane coordinates.



SAMPLE MAP SHOWING SEGMENT IDENTIFICATION  
FOR BLOCK 101

	BLOCK LEFT	BLOCK RT	FROM X-Y		TO X-Y	
(1)	101		+03759	+04222	+03900	+04250
(2)	101		+03910	+04260	+03915	+04500
(8)	101	109	+03937	+04522	+03777	+04502
(REV) (7)	101	102	+03766	+04509	+03733	+04251
(4)	102		+03688	+04292	+03710	+04262
(7)	102	101	+03733	+04251	+03766	+03509
(5)	102	110	+03773	+04503	+03523	+04492

BLOCK CHAINING TECHNIQUE USED BY THE  
CHAIN PROGRAM FOR SAMPLE MAP SHOWN

Figure 9. Block Chaining

### 6.3 Program

CHAIN makes use of block boundary segments digitized as single lines separating two units left and right. The program checks for a complete set of boundary elements surrounding each block and splits the boundary segments into left and right corresponding to their respective blocks. It then transforms the block file coordinates from a table coordinate file to a standard geographic coordinate system, maintaining scale and table rotation limit checks.

The CHAIN Program is written in three phases.

#### Phase I:

Determines, for each census tract, the geographic coordinates of the digitized X-Y origin, map scale, rotating angle to convert digitized coordinates to a standard reference system and tests to determine if control points exceed user specified tolerance.

#### Phase II:

Reverses and duplicates segments which are boundaries for a block coded with left and right geographic codes and produces a complete boundary file for each block.

#### Phase III:

Transforms the standard scale and rotation angle of the polygon produced in Phase II, and converts all records for each tract so they will conform to the geographic reference system established in Phase I.

Control coordinates must fall within accepted limits and blocks must chain, i. e., start and end on the same node, to be continued into Phase III, transformation. Failure of the file Phase I or II will generate an error message and stop processing.

6.3.1 Input. The control specifications, digitized and calculated control points, and digitized segment file are shown in Figure 10.

6.3.2 Output. Output includes a listing of the transformation solution, rotation and scaling factors, input digitized coordinates, listing and deck of transformed geographic coordinates, and any error messages that are invoked.



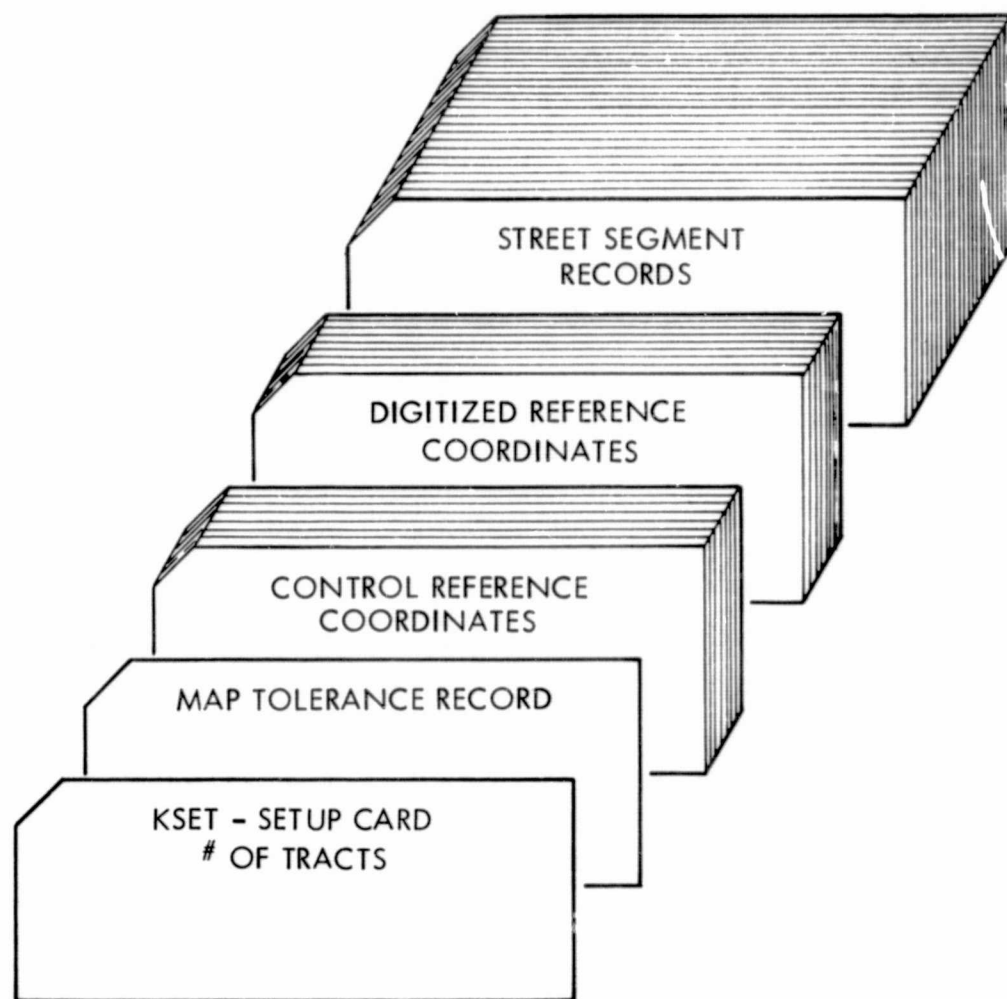


Figure 10. Deck Setup for Chain for Multiple Block Boundaries

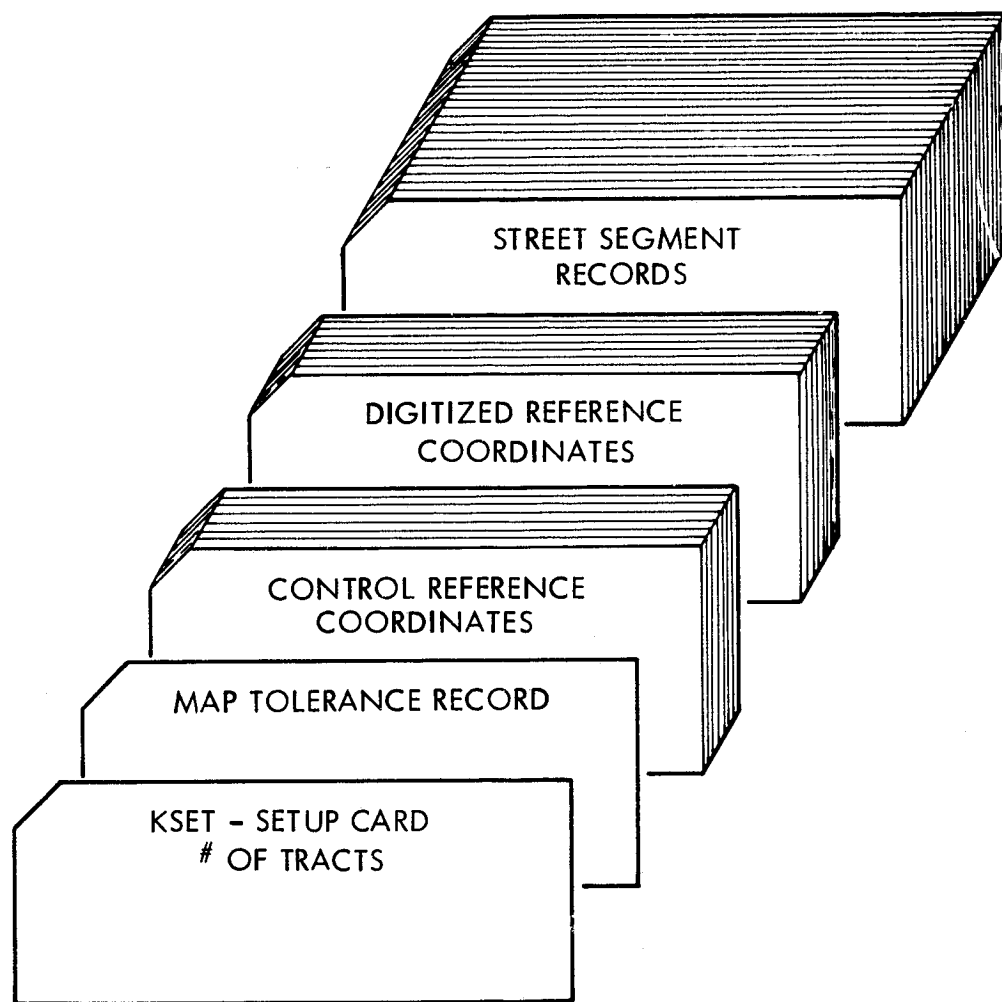


Figure 10. Deck Setup for Chain for Multiple Block Boundaries

## 6.4 Overall Specifications

Blocks are digitized as units in a tract, therefore geographic reference control points must be established per each tract. Each map contains one or more tracts.

COL 5

4  
KSET

## 6.5 Phase I Specifications

Phase I requires as input the following information: map tolerance record, control reference record, digitized reference record.

6.5.1 Map tolerance record establishes the limits in calibration of the digitized coordinate reference points and the overall parameter of the polygon to follow.

NNPOLY - Total number of blocks in a tract.

M - Number of control points provided as calculated and digitized reference records.

X0TOL, Y0TOL, THETOL, user specified tolerance levels in the solution of digitized origin (X0, Y0 in inches) and the rotation angle ( $\theta$  - in minutes of arc) between digitized map and accepted reference source.

COL	5	10	20	30	40
-----	---	----	----	----	----

```

      48      7      .045      .045      .500
NPOLY      M      XTOL      YOTOL      THETOL

```

6.5.2 Control reference points are calculated from an existent reference source by the user. They are given in tenths of a foot without a decimal point. In this case the control points are in California Zone 7 State Plane Coordinates, but any legitimate earth coordinate system is acceptable. One through M coordinates are coded with four or less pairs per card in the following format:

COL	9	18	27	36	ETC....	72
	<div style="border-left: 1px solid black; border-top: 1px solid black; padding-left: 10px; margin-top: 10px;"> 44349278      424712321  X(1)              Y(1)      X(2)      Y(2) </div>					

6.5.3 Digitized reference points are entered in the same order as the control reference points in thousands of an inch, without the decimal point. One through M coordinates are coded with six or less pairs per card in the following format:

COL	6	12	18	24	30	36	ETC...	72
	<div style="border-left: 1px solid black; border-top: 1px solid black; padding-left: 10px; margin-top: 10px;"> +24789      +12345  XD(1)      YD(1)      XD(2)      YD(2) </div>							

Upon completion of the Phase I, the user will receive a message as to the status of the job. The message indicates the calculated scaling factor and rotation angle.

## 6.6 Phase II Specifications

If Phase I is completed without errors, Phase II requires link records for each street segment in a census tract.

Street segment record. Each record contains the left and right census tract/block for a segment and the beginning and ending coordinates. If a segment is missing a tract/block right, the coordinates are coded but the right position is left blank as shown in Figure 11.

ITLEFT = Tract/block left of street segment (1 to MPOLY)

ITRITE = Tract/block right of street segment (1 to NPOLY)

ORIGINAL PAGE IS  
OF POOR QUALITY

6-8

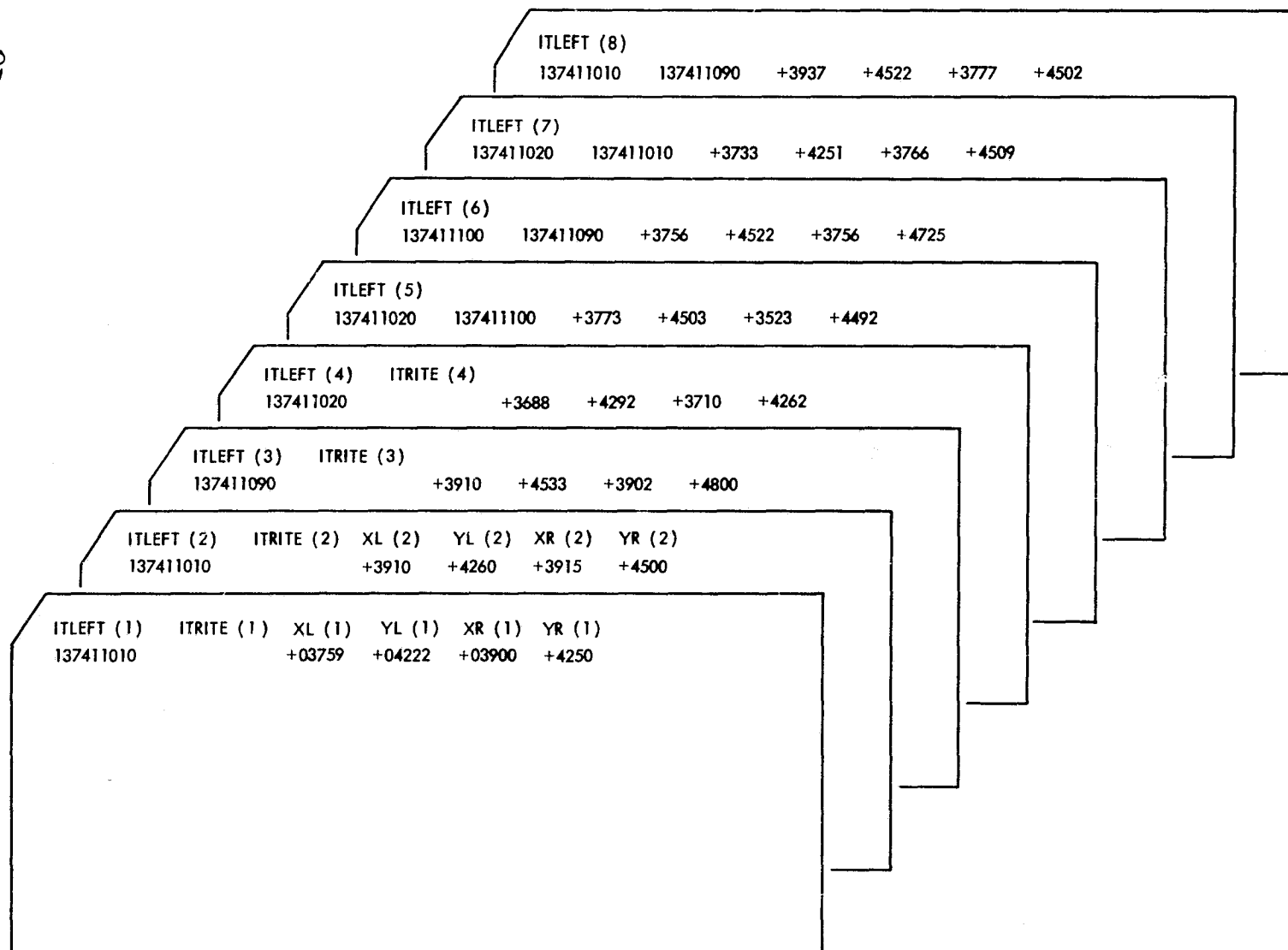


Figure 11. Sample Block Boundary File Layout

XL, YL = X and Y coordinates of the beginning node for a segment.

XR, YR = X and Y coordinate of the ending node for a segment.

COL	9	18	26	32	38	44
	137411020	137411100	+03773	+04503	+03523	+04492
	ITLEFT	ITRITE	XL	YL	XR	YR

## 7.0 DIME AREA CENTROID SYSTEM (DACS)

### 7.1 Introduction

The DIME Area Centroid System (DACS) was modified and adapted to the LUMIS project as an alternative to the CHAIN program to create a census tract/block polygon input to PIOS (Polygon Intersect Overlay System) from the DIME file. The DACS program translates a segment record structure used in the DIME file to a single sided polygon outline file which is used by PIOS.

DACS was originally developed by the Census Use Study, a Division of the Bureau of the Census. It was designed to produce edited and reformatted subfiles from the original DIME format. The subfiles can be used in computer mapping, area calculations, and adjacency problems. The editing feature allows the user to correct the file at the same time.

This documentation describes the basic program logic, and data input and output files in the program. This version includes corrections and modifications made by the Jet Propulsion Laboratory to create a PIOS input file. The program and modifications are written in FORTRAN IV for use with an IBM 360/370 computer, but can be adapted to other systems.

### 7.2 Methodology

DACS is written in three phases. These phases deal with the creating of a single sided segment intermediate subfile, a sort, and editing and reformatting an output file.

An intermediate subfile is created from all boundary segments for a selected region: census tract, census block group, or block which is identified by having two different right and left regions. The data for each boundary is written to the subfile twice, once unaltered as the boundary to the first polygon, and then with the beginning and ending node reversed to create a boundary for the second polygon. The final file contains separate records for each left boundary using counterclockwise orientation, and true records for the right boundary.

In Figure 12 segment A to B has a left side with census tract 1011.00 block 101 and a right tract of 1013.00 block 101. The first segment is written to the subfile as tract 1011.00 block 101 and coordinate for nodes A to B. The second segment is for tract 1013.00 coordinates from B to A, putting that tract on the left side with relation to this X-Y coordinate.

In the sort step, the subfile is sorted from the leftmost region in preparation for the third step.

In the third step, the program edits the file, performs any necessary calculations and reformats the output. All the boundary lines for one region are read in and stored. The array is then passed to the chaining edit program to determine if the segments are for one or more closed loops. If the segments do not close, the user is notified of the condition by a printed message and all calculations are skipped, (see Figure 13A). If the segments form more than one closed loop (Figure 13B), a message indicating the condition is printed, but in this case calculations are not allowed to proceed.

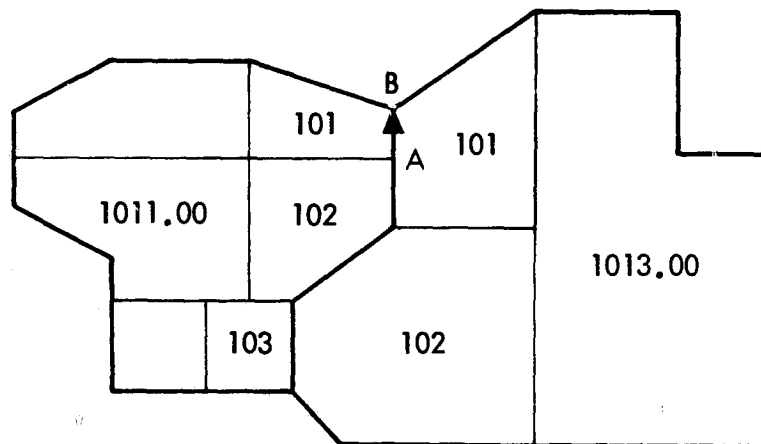
Regions accepted by the chaining test are passed to routines to calculate the centroid coordinates and the area, build lists of adjacent regions, or reformat the segments for use in mapping and overlay programs. The complete documentation for area, adjacency and centroid routines is available through the Census Bureau.

The subroutine to reformat the DIME file for PIOS input is appended to the third step of DACS. The subroutine first modifies the tract and block identifiers by removing the case to form the description record. Minimum and maximum coordinates are determined through a simple logical test, at the same time counting the number of coordinate pairs to form the second polygon record, minimum and maximum coordinates. Finally the X, Y coordinates which describe the polygon are written to a subfile in sets of four as specified for PIOS input.

### 7.3 Program

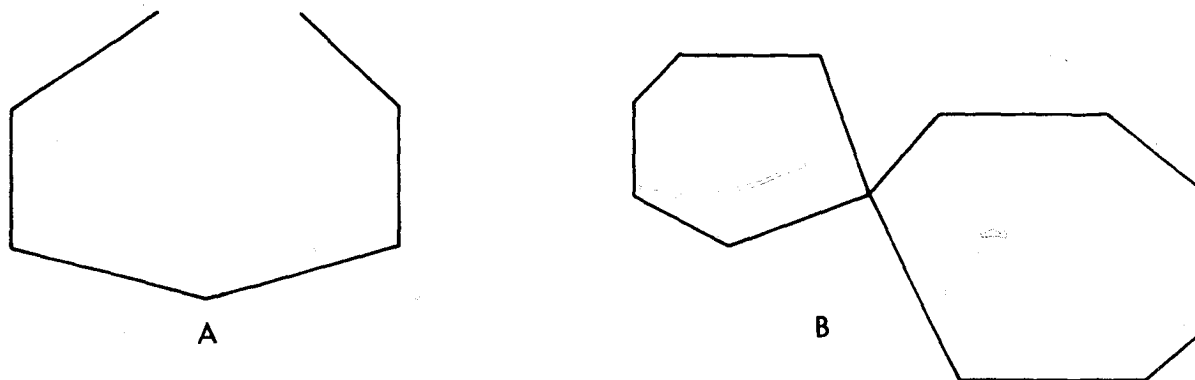
The DIME Area Centroid System (DACS) consists of two programs designed to calculate areas and locate centroids of user-specified blocks, tracts, block groups, or other local areas defined in a DIME File. For each region





Map showing boundary segment A-B dividing census tract 1011.00 and 1013.00. Each segment in the DIME file is identified by both the left and the right geocodes making each polygon independent. DACS can then convert DIME files into complete polygon boundary description for PIOS input.

Figure 12. Boundary Segment Dividing Census Tract



Examples of two possible polygons edited by the DACS program.  
 A. A block which will not chain.  
 B. A block which chains, but with more than one closed polygon.

Figure 13. DACS Edited Polygons

considered, DACS produces, at user request, a tape listing X and Y coordinates of the boundary segment endpoints forming the polygon vertices and/or type and printed listings of adjacent regions. The centroids and boundary coordinates are useful as input to computer mapping programs such as GRIDS and SYMAP, and as input to polygon overlay programs, such as PIOS.

Errors and anomalies in the area and centroid values, identified by the program, are helpful in locating coding errors in the input file.

DACS is written as a preprocessor, sort and calculation program. These are used in three distinct phases.

Phase I:

The first program, DACS presort, uses an input DIME file tape to prepare an intermediate file; the intermediate file passes the control parameters and the selected boundary segments will be used in the calculation program; thus two programs can be thought of as phases of the same program.

Phase II:

The second phase sorts the output from the preprocessor so that all the segments from a region will be in one location on the file.

Phase III:

The third phase performs the actual computations, edits the file and writes any output files. It reads the sorted intermediate tape and stores segment data for a region internally. When the first record for a different region is encountered, the accumulated data is passed to subroutine CHAIN.

Subroutine CHAIN checks to see if the region's boundary segments form one or more closed loops. If they do not, a message to notify the user of the condition is printed, and calculations are skipped. If they form more than one closed loop, another message is printed; however, in this case, calculation is allowed to proceed.

Data for a valid region is used in subroutine CALC, which computes the area and centroid coordinates. Subroutine POLYPT then determines whether or not the centroid lies within the region boundary. If it lies outside

the boundary, an appropriate message is printed, and the centroid X and Y coordinates are set equal to those of the nearest boundary segment node. Subroutine REFMT reformats the data into a file which can be used by PIOS.

There are a maximum of four output tapes created by DACS. The area and centroid data, with identifying region number, are written onto the first tape, a mapping file onto a second tape, an adjacency file is written onto a third, and PIOS formatted file is written to a fourth tape.

7.3.1 Input. The preprocessor program uses a standard 300 character formatted DIME file or a binary file depending on the user specification and the control parameters. The control parameters are read as cards and passed to the second program as the first input record.

7.3.2 Output. Four output files can be produced by DACS. A file of polygon areas and centroids is created in each run as a standard output. The remaining files are optional, and a mapping file, an adjacency file and a PIOS formatted file.

The mapping file consists of endpoints or vertices of each region's boundary segments. The record to describe a polygon is written in succession and in a format usable by SYMAP.

The adjacency file lists all regions bordering the region analyzed; one adjacent region is listed per record, and all records pertaining to a region occur in succession.

The PIOS compatible file is an 80 character file similar to the mapping file, but is formatted differently. Each polygon has a description record, minimum and maximum X, Y coordinates and the vertices written as four per record.

DACS will also list the polygon's area, and its centroid coordinates -- or an adjacency list at the user's request.

The user is continuously notified of error conditions. Possible error conditions include improperly defined regions and anomalies in centroid location, as well as errors in the coding of the original DIME file.

Possible file error conditions can occur because:

- a) The polygon has more than 2000 segments.
- b) The segments form more than one polygon.
- c) The segments do not close.
- d) A segment on the file must be reversed before proceeding with processing.
- e) The centroid of the polygon falls outside the polygon and must be adjusted.

The polygon only stops processing in case a) or c); in all other cases a warning is printed and processing continues.

#### 7.4 Phase I Specifications:

DACS preprocessor, Phase I, has two input types: a card deck and the tape file. The card deck passes control specifications to the program to determine the input and output selections, and the area of study. The first card, the problem card, selects the user options. This is followed by a series of cards which selects the required census tracts, blocks, etc. contained in the study area.

If SELECT cards are included, the program performs calculations only for the regions identified by the cards; if REJECT cards are included, it considers all regions except those identified. If neither SELECT nor REJECT cards are used, the program calculates areas and centroids for all regions in the file.

Input: Profile Cards

File — Type of input file

1-6 character or binary

KALL — Type of region to be considered

11-14 TRCS for tracks

BLKS for blocks

RGNS for regions identified by area codes

BGPS for block groups; if left blank, the program assumes tracts are desired.

CODE - Type cards used to define the study area,

19-24 selection or rejection

SELECT if selection cards follow

REJECT if rejection cards follow

Blank if neither selection nor rejection cards are included.

ELECT - Type of region specified on selection or

27-30 rejection cards to follow.

TRCS for tracts

BLKS for blocks

RGNS for regions identified by area codes

BGRP for block groups

Blank if no cards are included.

AREA - Units in which area is to be calculated

46-49 SQFT for square feet

SQMI for square miles

ACRE for acres

If left blank, the program will calculate areas in square feet

OPTION - Optional output file specified.

59 1 if printout of areas and centroids desired

0 if printout not desired

60 1 if tape with SYMAP boundary coordinates desired

0 if tape not desired

61 1 if "adjacency list" is to be produced (tape and printer)

0 if "adjacency list" not desired.

62 1 if tape with PIOS boundary coordinate desired

0 if tape not desired.

TITLE - Title that will appear at the top of each page of

69-80 printed output.

COL	6	14	24	30	49	62	69	80
-----	---	----	----	----	----	----	----	----

BINARY TRCS SELECT TRCS SQFT 0100 CENSUS----

## SELECTION/REJECTION CARDS. ONE CARD

For each tract, block, or region to be selected or rejected.

If regions specified are tracts:

1-3      Blank  
4-9      Tract number

If regions specified are blocks:

1-6      Tract number  
7-9      Block number

If regions specified are identified by area code:

1-6      Blank  
7-9      Area Code

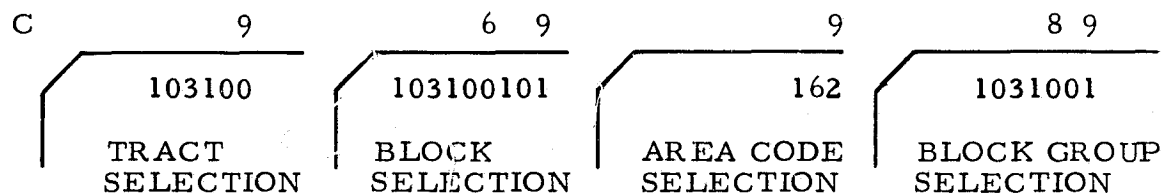
If regions specified are block groups

1-2      Blank  
3-8      Tract number  
9        Block group number

### Restrictions and Assumptions

Number of selection/rejection cards = 2000.

Number of boundary segments for any region specified = 2000



Input: tape

A standard Census Bureau tape is the second input to DACS preprocessor. The program will accept either a standard 300 byte character file or a 324 byte binary file as input.

Output: tape

The output from DACS preprocessor is an intermediate tape file with information about output options and a description of all boundary segments selected from the original ACG-DIME File.

The first record of the tape is the control record.

Format is as follows:

<u>Columns</u>	<u>Format</u>	<u>Data Item</u>
19-29	F11.0	Multiplier, to convert calculated area to proper units.
30-33	A4	Units (SQFT, SQMI, or ACRE)
34-37	A4	Region type (TRCS, BLKS, BGRP, or RGNS)
38-41	4I1	Code to specify selection of output options
42-53	3A4	Title, to appear on output page

The remainder of the records describe the boundary segments.

Boundary segments are those having different "right" and "left" regions. Data for each such segment in the original file is written twice on the intermediate tape—once with its original orientation, once with orientation reversed (i. e., with "left" and "right," "from" and "to" designations switched).

Format for each segment record is as follows:

<u>Columns</u>	<u>Format</u>	<u>Data Item</u>
1-9	I9	"Left" region
10-18	I9	"Right" region
19-27	I9	Segment serial number
28-37	I9	"From" node
38-47	I9	"To" node
48-57	I9	"From" node x coordinate
58-67	I9	"From" node y coordinate
67 bytes		Record length

#### 7.5 Phase II Specifications:

Before it is used as input to the DACS calculation program, the tape must be sorted by "left" region (Columns 1-9).

## 7.6 Phase III Specifications:

The calculation program requires very little user involvement with the exception of JCL requirements.

Input: The calculation program accepts the output file from the sort step directly without modification.

Output: The program has one output tape that is standard with three additional optional tapes.

Area-centroid tape is a file with the polygon area and its centroid produced on each run. The record format is as follows:

<u>Columns</u>	<u>Format</u>	<u>Data Item</u>
1-10	I10	The key used in the selection of block, track, block group, block or right justified block group is preceded by tract number or local area.
11-30	F20.5	Area (in user-specified units)
31-40	I10	Centroid "x" coordinate (map miles)
41-50	I10	Centroid "y" coordinate (map miles)

Record length = 50 bytes.

The "computer mapping" option produces a tape with coordinates of end point nodes of each region's boundary segments, all pertaining to a particular region occurring in succession. Output is written to File 11.

Record format as follows:

<u>Columns</u>	<u>Format</u>	<u>Data Item</u>
11-20	F10.0	"From" node y coordinate.
21-30	F10.0	"From" node x coordinate.
70-76	I7	Number of left-hand tract, block, block group, or local area.
77-80	I3	Sequence number of segment in the list of boundary segments for the region.

Record length = 80 bytes.



The "adjacency list" tape option produces a tape containing a list of all regions (blocks, tracts, block groups, or special local areas) bordering on each region considered. One adjacent region is listed per record, and all records pertaining to a particular region occur in succeeding records. Output is written to File 12.

Record format is as follows:

<u>Columns</u>	<u>Format</u>	<u>Data Item</u>
1-15	I15	Region number
15-30	I15	Adjacent region number

Record length = 30 bytes.

The "PIOS data file" tape has three record types; description record, minimum maximum coordinates, and polygon boundary record. Output is written to File 13, Record format is as follows:

Record Type 1

<u>Columns</u>	<u>Format</u>	<u>Data Item</u>
1-5	I5	Census tract ID deleting first digit of suffix.
6-10	I5	Block ID deleting first digit of suffix.
11-13	A3	Block ID - first three numbers
14-17	I4	Sequence number.
18-19	I2	Polygon type

Record Type 2

<u>Columns</u>	<u>Format</u>	<u>Data Item</u>
1-9	I9	Minimum x coordinate
10-18	I9	Minimum y coordinate
19-27	I9	Maximum x coordinate
28-36	I9	Maximum y coordinate
37-41	I5	Number of coordinate pair describing polygon.

### Record Types

<u>Columns</u>	<u>Format</u>	<u>Data Item</u>
1-9	I9	X Value
10-18	I9	Y Value
19-27	I9	X Value
28-36	I9	Y Value
37-45	I9	X Value
46-54	I9	Y Value
55-63	I9	X Value
64-72	I9	Y Value

### Printed Output

For the properly defined regions considered, the user may request a printout of area and centroid coordinates and/or a polygon adjacency list.

## 8.0 POLYGON INTERSECTION OVERLAY SYSTEM (PIOS)

### 8.1 Introduction:

The polygon overlay program is part of the PIOS (Polygon Intersection Overlay System); it uses output data from the ROTATE and CHAIN programs to compute statistics for major and minor polygon intersections. PIOS was originally developed for the San Diego Comprehensive Planning Organization (CPO by Environmental Systems Research Institute (ESRI) of Redlands, California). The system was designed and implemented in 1971 under a contract from CPO to ESRI to digitize soils polygons for San Diego County and then quantify soils type within Traffic Analysis Zones (TAZ), which are nominal statistical areas used for planning purposes by CPO.

This documentation describes the program logic and the data files of the Polygon Information Overlay System program Version 75, (PIOS-75). This version includes the modifications which the Jet Propulsion Laboratory has made to PIOS II, in order to accommodate a specific land use polygon overlay problem. This modified version retains PIOS II, which was created for the CPO in San Diego. PIOS-75 will run on either the Univac 1108, or the IBM 360/370 computers.

### 8.2 Methodology:

PIOS numerically overlays any general set of ordinarily defined polygons. These are identified as a major and a minor polygon, and their overlay produces a residual polygon of their intersections. This residual polygon can be overlaid with a third polygon, and so forth, each overlay producing a residual polygon of common intersection. This overlay system is conceptually similar to a "cookie cutter."

The census blocks are the major polygons in the overlay technique employed in LUMIS. In the Los Angeles test eight digitized ordinal maps became the minor polygons. Major polygons, after processing with the polygon overlay software, contained the information overlaid from the minor polygon. These minor polygons while proposed were never created from land use and natural resource maps, slope maps, elevation maps, geology maps, landslide and fault maps, soil maps, and air pollution indices.

PIOS reads all minor (land use) polygons and stores their approximate location, i.e., the minimum and maximum boundaries or window. The window technique allows complex polygons to be treated as boxes which are sorted and manipulated in their general format until a more accurate definition is required. This technique permits more efficient computational analysis of a complex polygon file.

Having windowed the minor polygon file and stored it in a temporary direct access file, the program reads the major polygon file (census blocks). Before any analysis of overlapping polygon takes place the program breaks up the major polygon into strips (see Figure 14). New points are generated at strip boundaries so that each strip is wholly contained.

The logic of this stripping technique is not unique, but it does provide for substantial increase in program efficiency. Briefly the strips reduce the amount of core storage required by the program by only retaining that portion of the polygon that is being analyzed for intersection at one time. The number of strips is determined by the tradeoff between efficiency gained in the processing, and the core storage requirements necessary to run the program. As the complexity of polygons increases, the efficiency gained by increasing the number of strips also increases. The point-in-polygon technique uses directed line segments in the positive X-direction of slope zero; line segments so directed will not cross over onto another strip. For the LUMIS test the major polygons were broken down into 16 strips of equal height.

After the major polygon (census block) has been stripped a minor polygon (land use) is read in to compute the intersection points, if any. Each point in the minor polygon is tested for being in or out of the major polygon.

If the minor polygon is determined to be inside the major polygon, there are three possible options.

Case 1 - The land use polygon is completely and totally contained within the census tract polygon.

Case 2 - The census tract polygon is completely contained within the land use polygon.

Case 3 - Part of the land use polygon resides within the census tract polygon, and a residual polygon is formed.

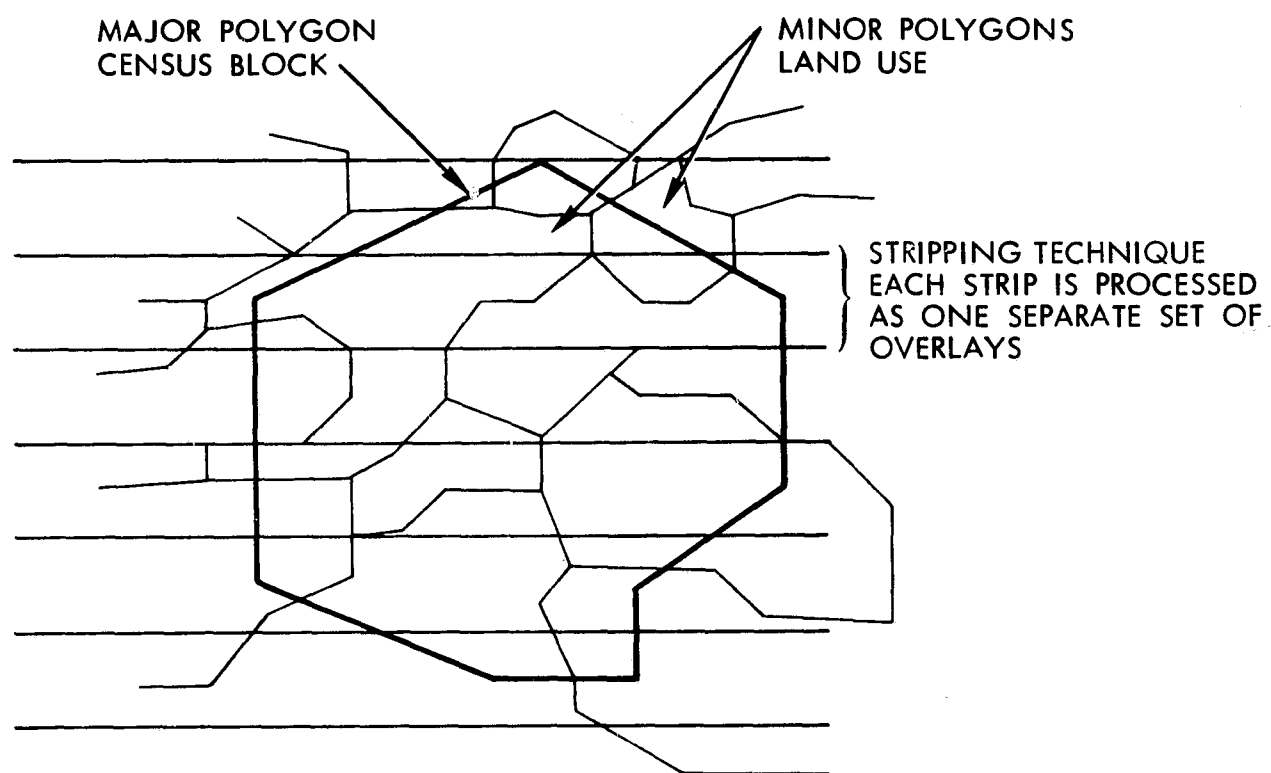


Figure 14. PIOS Stripping Method

In Case 1, PIOS-75 writes the entire land use polygon to the output file. In Case 2, the entire census tract polygon is written to the output file. In Case 3, PIOS-75 effectively cuts off and saves the part of the land use polygon which resides within the census tract. By creating two residual polygons from the original minor polygon, Figure 15 geographically displays Cases 1-3.

In Case 3, each segment of a minor polygon must be analyzed individually for being either completely inside, completely outside, or intersecting the major polygon. If the segment intersects the major polygon, it is entering or leaving. This is graphically pictured in Figure 16.

Four possible actions taken are tabulated below:

<u>Example</u>	<u>Action Taken</u>
A to B	Throw away point A. Do not consider segment as part of polygon.
B to C	Find intersection B'. Determine direction on major polygon. Follow major polygon until it goes out of minor polygon F'.
C to D	Add points to the line segment defining the new polygon being calculated.
D to E	
E to F	
F to G	Find intersection F'. Determine direction on major polygon. Follow major polygon until it goes out of minor polygon B'.

After selecting one of these actions, a new polygon is developed and subsequently computed for area.

PIOS-75 will examine each land use polygon in sequential order while looking for a match. If after checking all land use polygons no match is found, the main program will request the next census tract to be read in for the same review. If it finds no match between any land use polygon and census tract polygon, the program will terminate.

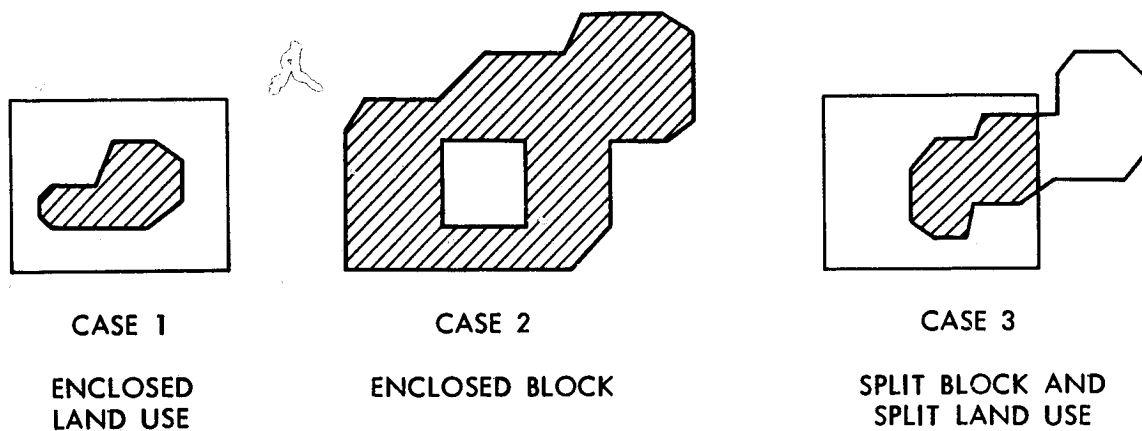


Figure 15. Example of Alternative Block and Land Use Intersections

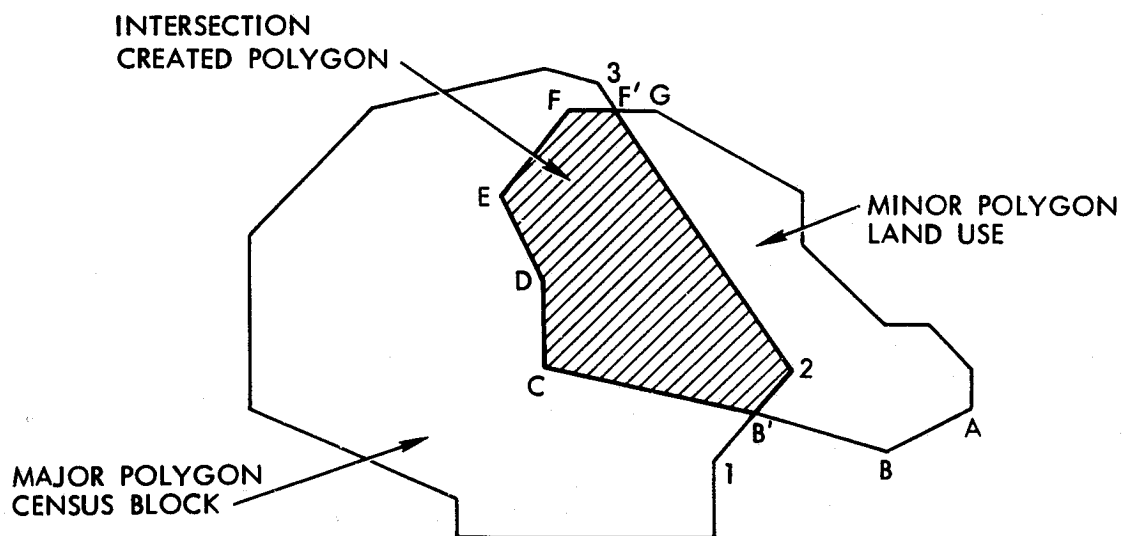


Figure 16. An example of identifying unique polygons created by intersecting blocks and land use

As part of the JPL improvements to PIOS, the program lists census tracts and blocks, the residual land use area, and percentage each makes of the entire tract per each run. The listing also includes those tracts for which no overlay was possible as well as those for which the entire census tract fell into one minor polygon. Further abstractions or summaries can be derived from the output listing to meet the users needs.

The summary can be used as a preliminary means of checking for errors in the PIOS run. A better means for error checking is to sum all the residual areas and check the total against the original polygon as illustrated in Figure 17. A second supplemental program is provided for error checking. The second program uses as input the original minor polygon file (12) and the residual polygon file (14).

The residual file must first be sorted by land use map and code, polygon number, census tract and block. The program will aggregate and list the residual polygon coded for a land use map and type and number. If the entire land use was overlaid, the listings should show the polygon as 100% overlaid by blocks. In case of partial overlay, the results can be used to estimate the amount of coverage.

### 8.3 Program:

PIOS is used to determine the common area of two overlapping polygons. The program will overlay one or more "major" polygons onto the "minor" polygon file. Each major polygon is compared with each minor polygon for possible overlap. Both the major and minor polygons have a predefined "window" associated with them in the form of minimum and maximum X and Y coordinates. If the major and minor "windows" overlap, then the program uses a specialized version of the relatively common point-in-polygon technique to determine if any overlap actually exists. If two polygons do overlap, a polygon representing the overlap area is structured using the existing coordinates of the major and minor polygons. Once the structuring process is complete, the area of the new polygon is calculated and sorted in a "residual" polygon file for later use in tabular or graphic report generating programs. This process is illustrated in Figure 18.



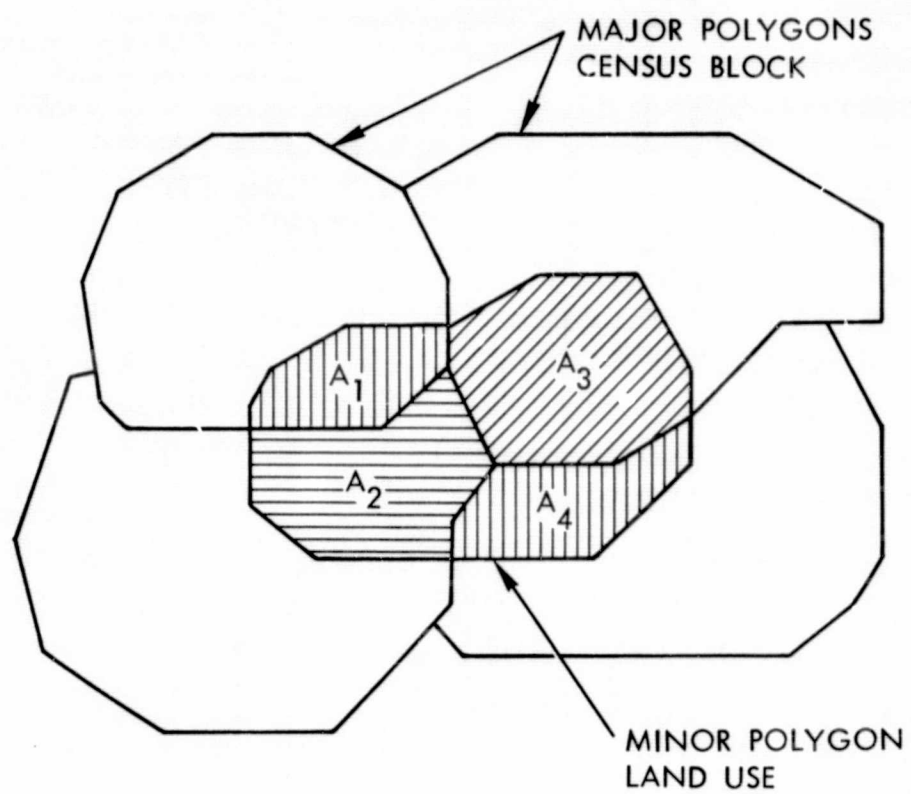


Figure 17. Reconstruction of Minor Polygons from Multiple Intersection Polygons ( $A_1 + A_2 + A_3 + A_4$ )

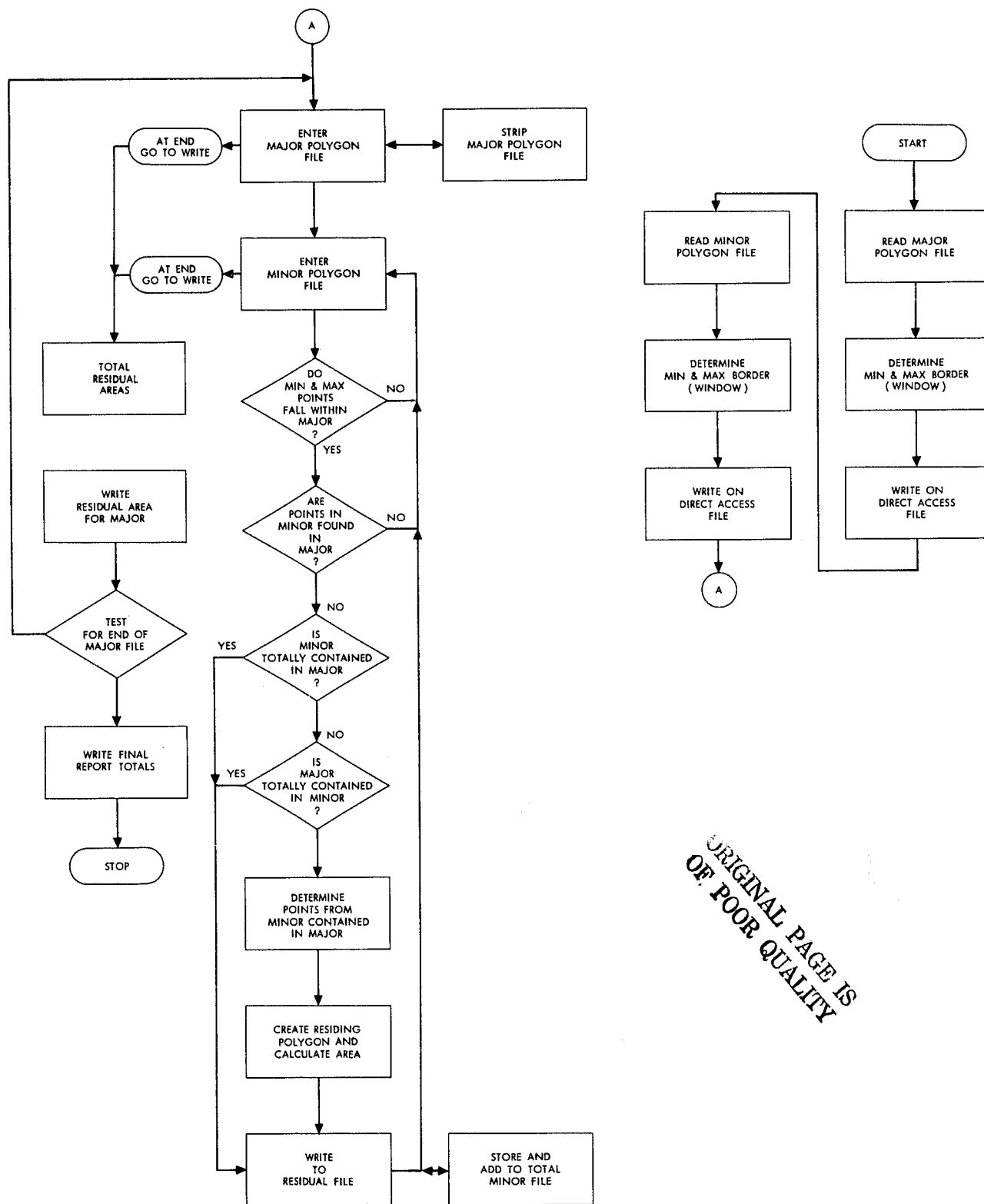


Figure 18. Flow Chart PIOS Program

The program is written as a series of FORTRAN IV subroutines invoked by the main program DRIVER.

Subroutine POLYRD (polygon read) is used to read each of the polygons in from the external file storage medium. The subroutine makes use of random-access techniques for reading both the major and minor polygon files. It also performs the function of calculating centroids from the coordinate arrays of simple and containment type polygons.

The function LEVEL is used to provide level indices for the polygon overlay procedure. Major polygons are "stripped" into 16 horizontal sections or levels. When the points in the minor polygon boundary are submitted for the point-in-polygon examination, the levels of the major polygon could potentially contain the minor point. Then, only the line segments which make up that level are used by the PIP subroutine.

Subroutine PIP (point-in-polygon) determines whether a point on the minor polygon boundary is located inside or outside of the major polygon boundary. Establishing this relationship for each of the minor polygon vertices is the foundation of the overlay process.

Subroutine STPSUB (step subroutine) is used in conjunction with the residual structuring process. In cases where the direction of digitizing is not known and an intersection of major and minor boundaries is encountered, this subroutine provides an indication of the direction to "step" next.

Subroutine AREAOF is called by subroutine POLYRD to determine the area of a polygon, using the cross product. It also determines if the polygon has been digitized in the correct clockwise direction by the sign of the resulting area. If the polygon was digitized in the wrong counterclockwise direction, the polygon is reversed.

#### 8.4 Input Files:

The input files to the polygon overlay program are both in the same format. The files are differentiated by the program in that one is considered as the major file, the other as the minor file. The choice of which is the major file is the users'. The major polygons are usually the larger polygons, and the minor polygons would normally be those that are wholly or partially

contained within the major polygon. The major polygon file is identified as File 11 and the minor polygon file is File 12. Polygons from File 12 are matched up with, or overlaid upon, data from File 11. In the LUMIS project the census block was the major polygon file and land use was the minor polygon file. Figure 19 illustrates the deck setup for PIOS.

PIOS will process both simple and complex polygons. "Doughnut holes" within complex polygons will be written to the output file with negative acreages. This facilitates later summation processes and tabular reporting requirements. The only limitation on input files is that if a minor polygon file contains more than 1,000 polygons, only the first 1,000 will be used in the overlay process. Larger files can be processed by enlarging the internal storage arrays in the polygon overlay program, which increases the core storage requirements.

#### 8.5 Output Files:

The three files created by the PIOS are similar in content and somewhat redundant: The residual file, File 14, is a polygon file that contains the areas that were found to be common between the major and minor polygons. Each polygon contains the tract and block number of the major polygon and the land use code, polygon number and type code of the minor polygon, the calculated area of the polygon (in acres) and the coordinates of the polygon vertices.

The second output file, File 13, is a condensed version of the residual file. It contains the same information for each polygon as the residual file but the boundary coordinates are excluded. This file is more efficiently used in tabular summation programs and reporting applications than the residual file.

The third output file, File 15, is a condensed version of the census tract file. It contains the census tract and block identification and total area of census blocks.

8.5.1 File 11 - Major polygon file. It is a direct access file with census tract and block records. The file contains records 80 characters in length. Each census block is made up of three record types: a descriptor record, a MIN/MAX record, and the boundary points.

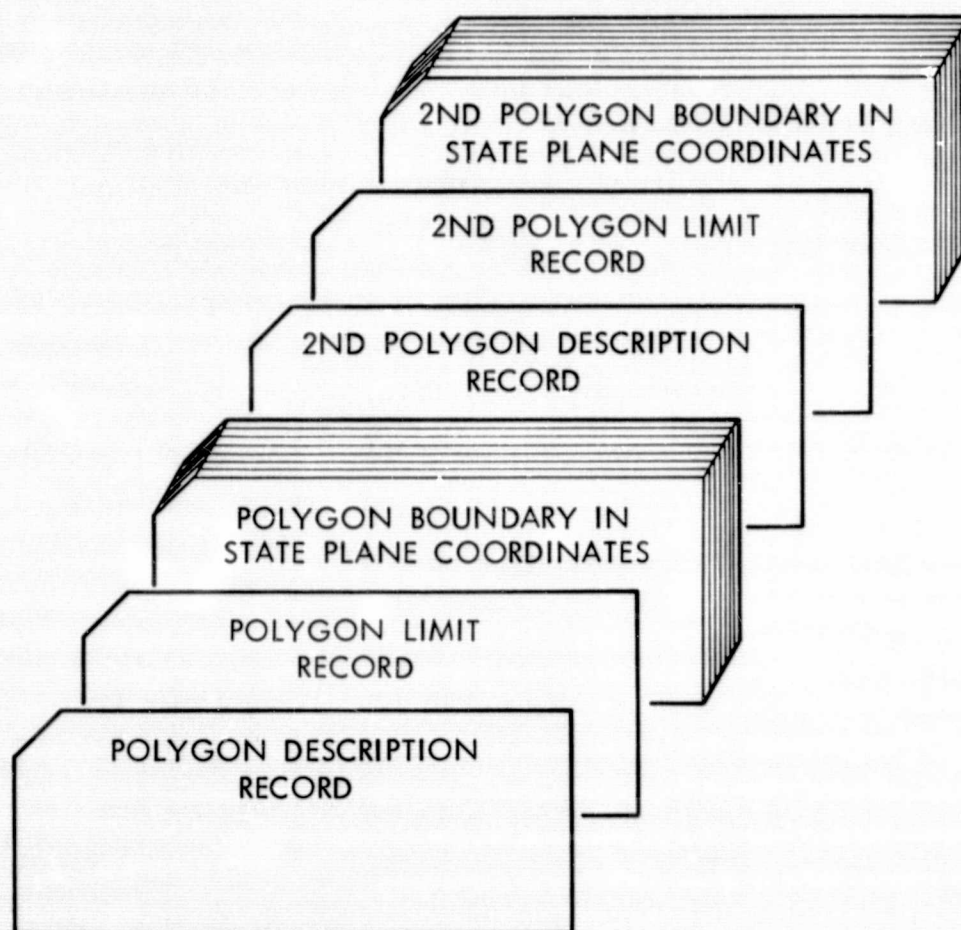


Figure 19. Major or Minor Polygon File Required by PIOS

The major polygon descriptor record contains basic identification information to describe the minimum and maximum points and the boundary record to follow. The following data names are used by the program:

- MAPNO: The census tract identification number is the census tract number with the first number of the suffix deleted. Since this number is always zero, it can be deleted to save space in later processing.
- MAPOL 1 (2): The block number identification number is the  
or census block number with the first number of the suffix  
LUCODE (2): deleted. As in the case of the census tract, this number can be deleted without losing information.
- LUCODE (1): The block number is repeated in this space dropping the entire suffix. This number serves as a filler in processing the census tract file, but is important in processing the minor file.
- LUCODE (3): The map sequence number denotes the sequence in relation to all the census tracts coded in a map.
- MAPOL 1 (2): The polygon-type code describes the character of each polygon. A normal polygon is coded 10. Doughnut polygons, (polygons within polygons) are coded by numbers larger than 10, starting with 11. Two mutually exclusive doughnut holes are coded 11, while nested polygons follow in order: 11, 12, --. All digitized doughnut holes must follow the parent.

COL        5       10       13       17       19

13511	1010	101	2	10
-------	------	-----	---	----

8.5.2 File 12 — Minor polygon file. It is a direct access with the land use or ordinal polygons. The file contains records 80 characters in length. Each land use polygon is described by three record types similar to the census tract input with minor modifications and different data names.

The record 1 minor polygon descriptor record contains basic identification information for the minimum and maximum points and boundary records to follow.

The following data names are used in the Program:

SPMAP: The identification number of the ordinal/land use map that is being processed.

SPPOLY (1): The total numbers of polygons coded using a map or identification number.

SPCODE:

SPCODE (1): The identification code identifies the land use type as defined in the land use map legend.

SPCODE (3): The sequence number denotes the sequence in relation to the total number of polygons coded in a map.

SPPOLY (2): The polygon-type code describes the character of each polygon. A normal polygon is coded 10. Doughnut polygons (polygons within polygons) are coded by numbers larger than 10, starting with 11. Two mutually exclusive doughnut holes are coded 11, while nested polygons follow in order: 11, 12, --. All digitized doughnut holes must follow the parent.

COL        5   10    13   17   19

17401   80   X03   60   10

8.5.3 Record 2 major/minor - min/max record. The Min/Max Record describes the extreme points of the polygon or the Minimum X and Y coordinate and the Maximum X and Y coordinate, and the number of coordinate pairs which will follow to describe the polygon's perimeter.

<u>Columnn</u>	<u>Format</u>	<u>Data Item</u>
1-9	I9	xmin - The minimum x coordinate value of the polygon
10-18	I9	ymin - The minimum y coordinate value of the polygon
19-27	I9	xmax - The maximum x coordinate value of the polygon
28-36	I9	ymax - The maximum y coordinate value of the polygon
37-41	I5	n - The number of coordinate pairs describing the polygon

8.5.4 Record 3 to 3+N/4 pairs major/minor record. Polygon boundary record (S) - This set of records describes the polygon boundaries in terms of x and y coordinate pairs starting with x, y to x, y

<u>Columnn</u>	<u>Format</u>	<u>Data Item</u>
1-9	I9	x value
10-18	I9	y value
19-27	I9	x value
28-36	I9	y value
37-45	I9	x value
46-54	I9	y value
55-63	I9	x value
64-72	I9	y value

8.5.5 End of File records. A sequence of four 9's (9999) is recognized by the IBM 360/370 program as an end of file when located in bytes 1-4. A sequence of five 9's (99999) is recognized by the U1108 program as an end of file when located in bytes 1-5. The last two records of each direct access file (File 11, 12, 13 and 14) should contain these codes.



#### 8.6 Output:

File 13 contains the residual polygon produced by the PIOS Program. Each record is stored on a direct access device as a 37-character record. The record incorporates the residual polygon census tract identification with the land use codes and the polygon area. The record output is in the following format:

<u>Column</u>	<u>Format</u>	<u>Data Item</u>
1-5	I5	Census tract numbers of major polygon involved in the overlay.
6-10	I5	Block number of major polygon involved in the overlay.
11-15	I5	Land use map or minor polygon in overlay.
16-18	A3	Land use identification information code.
19-22	I4	Polygon sequence number or land use involved in the overlay.
23-37	F15.2	Area of the residual polygon file.

File 14 contains essentially the same descriptive information as File 13, the census tract and land use code in an 80-column format. It adds to this information a series of x and y coordinates that describe the residual polygon that is formed in the overlay process.

### 8.6.1 Record 1

<u>Column</u>	<u>Format</u>	<u>Data Item</u>
1-5	I5	Census Tract
6-10	I5	Census Block number
11-15	I5	Land use map
16-18	A3	Land use map identification code
19-22	I4	Sequence number of the land use polygon
23-37	F15.2	Area of the residual polygon
38-80		Blank

### Polygon Boundary Coordinate Pairs

### 8.6.2 Record 2 thru 2+N/4

<u>Column</u>	<u>Format</u>	<u>Data</u>	<u>Item</u>
1-9	F9.1	X <sub>1</sub>	Value
10-18	F9.1	Y <sub>1</sub>	Value
19-27	F9.1	X <sub>2</sub>	Value
28-36	F9.1	Y <sub>2</sub>	Value
37-45	F9.1	X <sub>3</sub>	Value
46-54	F9.1	Y <sub>3</sub>	Value
55-63	F9.1	X <sub>4</sub>	Value
64-72	F9.1	Y <sub>4</sub>	Value

File 15 is the second abbreviated file produced by PIOS. This file contains the same identification information found on the census tract descriptor record: tract, block, and sequence number.

Census tract area calculated in the PIOS Program is added forming the final data item on the record. The record is 29 characters in length and written on a direct access device in the following format:

<u>Column</u>	<u>Format</u>	<u>Data Item</u>
1-5	I5	Census tract number of major polygon
6-10	I5	Block number of major polygon
11-14	I4	Sequence number of block with census tract
15-29	F15.2	Area of the block

APPENDIX I  
SPCCAL PROGRAM LISTING

C STATE PLANE COORDINATE CALCULATION PROGRAM (SPCCAL)

```
5 FORMAT (I5)
10 FORMAT (I10,I5)
15 FORMAT (- GENERAL MAP NO.--,I10)
30 FORMAT (8F10.1)
32 FORMAT (- CONTROL POINT--,1X,I2)
35 FORMAT (4F10.3)
40 FORMAT (2F10.1)
45 FORMAT (-*****-)
50 FORMAT (- -)
70 FORMAT (- *****-)
80 FORMAT (- CONTROL POINT EXCEEDS ERROR FACTOR-)
90 FORMAT (2F10.1,4X,-ERROR-)
```

C INPUT NUMBER OF MAPS FOR WHICH THE CONTROL POINT SPC S ARE TO BE CALCULATED.

```
1 READ (5,5) NUMAP
DO 8 J=1,NUMAP
WRITE (6,70)
WRITE (6,50)
```

C INPUT THE MAP NUMBER (MAPNO) AND TOTAL NUMBER OF CONTROL POINTS (NCONPT)

```
READ (5,10) MAPNO, NCONPT
WRITE (6,15) MAPNO
WRITE (6,50)
DO 20 I=1, NCONPT
```

C THE X1, X2, X3, X4, Y1, Y2, Y3, Y4 SPC COORDINATES OF THE FOUR 2 1/2 MINUTE  
C LATITUDE-CONGITUDE INTERSECTIONS ARE READ IN.

```
READ (5,30) A,B,C,D,E,F,G,H
WRITE (6,32) I
WRITE (6,50)
```

C THE (SMALL LETTER X,Y) X1, X2, Y1, Y2 DIGITIZER MEASURED DISTANCES ARE READ

```
C O=X1
C P=X2
C Q=Y1
C R=Y2
```

```
READ (5,35) O,P,Q,R
O = O*2000.
P = P*2000.
Q = Q*2000.
R = R*2000.
```

ORIGINAL PAGE IS  
OF POOR QUALITY

```
X1 = A+O+(Q/(Q+R))*(C-A)
X2 = C+O-(R/(Q+R))*(C-A)
X3 = B-P+(Q/(Q+R))*(D-B)
X4 = D-P-(R/(Q+R))*(D-B)
Y1 = E+Q-(O/(O+P))*(E-F)
Y2 = F+Q+(P/(O+P))*(E-F)
Y3 = G-R-(O/(O+P))*(G-H)
Y4 = H-R+(P/(O+P))*(G-H)
Z1 = (X1+X2)/2.0
Z2 = (X3+X4)/2.0
Z3 = (Y1+Y2)/2.0
Z4 = (Y3+Y4)/2.0
Z5 = DABS(Z2-Z1)
Z6 = DABS(Z4-Z3)
IF (Z5 .GT. 30.0) GO TO 60
IF (Z6 .GT. 30.0) GO TO 60
T1 = (X1+X2+X3+X4)/4.0
T2 = (Y1+Y2+Y3+Y4)/4.0
WRITE (6,40) X1,Y1
WRITE (6,40) X2,Y2
WRITE (6,40) X3,Y3
WRITE (6,40) X4,Y4
WRITE (6,50)
WRITE (6,40) Z5,Z6
WRITE (6,50)
WRITE (6,40) T1,T2
IF (I .EQ. NCONPT) GO TO 43
WRITE (6,45)
43 WRITE (6,50)
GO TO 20
50 WRITE (6,80)
WRITE (6,50)
WRITE (6,40) X1,Y1
WRITE (6,40) X2,Y2
WRITE (6,40) X3,Y3
WRITE (6,40) X4,Y4
WRITE (6,50)
WRITE (6,90) Z5,Z6
IF (I .EQ. NCONPT) GO TO 44
WRITE (6,45)
44 WRITE (6,50)
20 CONTINUE
8 CONTINUE
STOP
END
```

ORIGINAL PAGE IS  
OF POOR QUALITY

C  
C \*\*\*\*\* SAMPLE DATA \*\*\*\*\*  
C

2  
17002 5  
4111065.3 4123669.9 4111102.5 4123700.8 4173149.8 4173121.4 4188312.8 4188284.5  
4.003 2.308 .160 7.432  
4111065.3 4123669.9 4111102.5 4123700.8 4173149.8 4173121.4 4188312.8 4188284.5  
5.566 .744 1.892 5.701  
4123638.9 4136249.6 4123669.9 4136274.4 4157958.5 4157935.3 4173121.4 4173098.3  
2.193 4.110 5.703 1.881  
4136249.6 4148860.4 4136274.4 4148878.9 4157935.3 4157917.3 4173098.3 4173080.3  
.780 5.527 7.075 .497  
4136249.6 4148860.4 4136274.4 4148878.9 4157935.3 4157917.3 4173098.3 4173080.3  
3.740 2.569 3.796 3.776  
17003 5  
4136249.6 4148860.4 4136274.4 4148878.9 4157935.3 4157917.3 4173098.3 4173080.3  
5.315 .998 4.287 3.288  
4148860.4 4161471.1 4148878.9 4161483.5 4157917.3 4157904.5 4173080.3 4173067.4  
.270 6.042 4.977 2.599  
4148860.4 4161471.1 4148878.9 4161483.5 4157917.3 4157904.5 4173080.3 4173067.4  
1.598 4.714 7.242 .333  
4148860.4 4161471.1 4148878.9 4161483.5 4157917.3 4157904.5 4173080.3 4173067.4  
1.930 4.381 3.705 3.869  
4148860.4 4161471.1 4148878.9 4161483.5 4157917.3 4157904.5 4173080.3 4173067.4  
4.244 2.067 4.188 3.387

A-I-4

APPENDIX II  
PLOTTER EDITING PROGRAM LISTING  
General Map Case



```

      INTEGER PTYPE,POLYNO
      DIMENSION XD(6000),YD(6000),STATEP(16),FRM(2)
      READ(5,500)JOF
      IF( JOF.EQ.999 .OR. JOF.EQ.777)GO TO 5
-----
      500 FORMAT(I4)
      600 FORMAT(1H ,10X,I4,I5)
      1000 FORMAT(2I5,A3,I4,I2)
      1002 FORMAT(4F6.3,I5)
      1004 FORMAT(12F6.3)
      1006 FORMAT(A5,2X,I3)
      1008 FORMAT(9X,I1)
      2000 FORMAT(1H0,10X'MAPNO MAPOL LUCODE POLYNO PTYPE'//
      $ 1H , 9X,I6, 15,5X,A3,4X,I4,5X,I2/1H ,9X,35('-'))
      2001 FORMAT(1H ,1X,'CONTROL POINTS',7(2F7.3,2X) )
      2002 FORMAT(1H ,10X,'XMIN YMIN XMAX YMAX NP=',I4,
      $ 5X,'**COUNTED**',I4,'**',
      $/1H ,6X,4F8.3)
      2003 FORMAT(1H ,2X,90('-'))
      2004 FORMAT(1H0,10X'DIGITIZED POINTS BY CARDS/LINE')
      2005 FORMAT(1H0,2X,'PLOTING')
      2006 FORMAT(1H ,10X,I4,'*', 5(2F7.3,2X) )
      2007 FORMAT(1H1,2X,A5,2X,A5,2X,A5/1H ,2X,'-----')
      2008 FORMAT(1H0,60('* '))
-----
C
      CALL PLOTS(0.0,20)
      CALL PLOT(0.0,-10.0,-3)
      CALL PLOT(0.0, 1.50,-3)
      IM = 3
      5 IFRAME = 0
      IF( JOF.EQ.999)GO TO 1
C
C
      2 READ(5,1006,FND=900)ITITLE,ISKIP2
      IF( JOF.EQ.777)GO TO 3
      CALL SYMBOL(0.0,-0.218, 0.1875,ITITLE, 0.0, 5)
      CALL NEWPEN(2)
      ENCODE(709,IADDR),IM
      709 FORMAT('AT-',I2)
      CALL SYMBOL( 0.950, -0.218, 0.1875, IADDR, 0.0, 5)
      IM = IM + 1
      3 IFRAME = 0
      IFRR=0
      XM = 999.000
      YM = 999.000
      XMX = -999.000
      YMX = -999.00
      WRITE(6,2007)ITITLE,ITITLE,ITITLE
      REWIND 25
C
C
      READ(5,1008,FRR=900,FND=900)ISTAPT
      IF( ISKIP2.EQ.999) READ(5,1004) (XD(N),N=1,14)
      IRN = ISTAPT*2
      READ(5,1004)(STATEP(I),I=1,IRN)
      DO 12 I = 1,IRN,2
      XM = MIN( XM,STATEP(I) )
      YM = MIN( YM,STATEP(I+1) )
      12 XMX = MAX(XMX,STATEP(I) )
      WRITE(6,2001)( STATEP(I),I=1,IRN)

```

ORIGINAL PAGE IS  
OF POOR QUALITY

WRITE(6,2003)

C  
C

```
1 IF( IERR.EQ.999)GO TO 9
  READ(5,1000,FND=9,ERR=900)MAPNO,MAPOL,LUCODE,POLYNO,PTYPE
  GO TO 7
6 READ(0,1000,FND=9,ERR=900)MAPNO,MAPOL,LUCODE,POLYNO,PTYPE
7 IERR = 999
  READ(5,1002          )XMIN,YMIN,XMAX,YMAX,NP
  XM = MIN(XM,XMIN)
  YM = MIN(YM,YMIN)
  XM = MAX(XM,XMAX)
  IF( JOE.EQ.999)WRITE(6,600)J,NP
  IFRAME = IFRAME + 1
```

C  
C

```
  J = 0
  DO 13 L = 1,6000,6
    M = L+5
    I = L
    READ(5,1004,ERR=14,END=15) ( XD(N),YD(N),N=I,M)
13 J = J + 6
    GO TO 16
14 IERR = 1
16 N = J-6
    DO 15 I=N,J
      IF( XD(I+1))15, ,15
      J = I
    GO TO 4
15 CONTINUE
```

C  
C

```
4 IF( JOE.EQ.999)GO TO 1
  IF( JOE.EQ.998)GO TO 8
  WRITE(6,2000)MAPNO,MAPOL,LUCODE,POLYNO,PTYPE
  WRITE(6,2002)NP,J,XMIN,YMIN,XMAX,YMAX
  WRITE(6,2004)
  NC = 0
  DO 21 N = 1,J,6
    M = N+5
    NC = NC + 1
    IF( M.GT.J)M=J
21 WRITE(6,2006)NC,( XD(NN),YD(NN),NN=N,M)
  WRITE(6,2008)
8 WRITE(25)J ,(XD(I),YD(I),I=1,J )
  IF( IERR.EQ.1)GO TO 6
  GO TO 1
```

C  
C

```
9 IF( JOE.EQ.999)GO TO 999
  WRITE(6,2005)
  IF(JOE.EQ.777)GO TO 2
  ENCODE(707,FRM)IFRAME
707 FORMAT(13,' POLYGONS')
  CALL SYMBOL( 0.0, -.469, 0.125, FRM, 0.0, 12)
  J = 0
  REWIND 25
  CALL NEWPEN(2)
  DO 11 I =1,IRN,2
11 CALL SYMBOL(STATEP(I)-XM, STATEP(I+1)-YM, 0.1875, 3, 0.0, -1)
  CALL NEWPEN(1)
```

ORIGINAL PAGE IS  
OF POOR QUALITY

```

10 READ(25)K ,(XD(I),YD(I),I=1,K )
   J = J + 1
   IPEN = 3
   DO 20 I = 1,K
     IF( XD(I) )20,20.
     CALL PLOT(XD(I)-XM, YD(I)-YM,IPEN)
20  IPEN = 2
     IF( IFRAME.EQ.J)GO TO 99
     GO TO 10
99  CALL PLOT(ABS(XMX-XM+2.00),0.0, -3)
     GO TO 2

```

C

```

900 IF( JOE.EQ.777)GO TO 999
     CALL FACTOR(0.5)
     CALL PLOT(2.5, 0.0, 999)
999 STOP
     FND

```

APPENDIX III  
PLOTTER EDITING PROGRAM LISTING  
MMS Map Case

```

      DIMENSION DATA(14)
      COMMON IONOF
1002 FORMAT(A3)
2000 FORMAT(1H0,130(' '))
      READ(5,1002) IONOF
      IF(IONOF .EQ. 'ON') CALL PLOTS
      IF(IONOF .EQ. 'OFF') CALL PLOTS(0.0,20)
      REWIND 27
C
      CALL TPLT
      REWIND 25
      REWIND 26
      WRITE(6,2000)
      CALL RNDRY
100 CONTINUE
      END FILE 27
      REWIND 27
      WRITE(6,2001)
2001 FORMAT(1H1)
105 CONTINUE
      READ(27,END=200) DATA
      WRITE(6,2002) DATA
      GO TO 105
200 CONTINUE
2002 FORMAT(1H ,14A6)
      CALL PLOT(0.,0.,999)
      END
"TEOR, IS TPLT,TPLT
      SUBROUTINE TPLT
      COMMON IONOF
      DIMENSION DATA(14)
C
1000 FORMAT(5X A3,7X A3,2X4F6.3)
1001 FORMAT(13A6,A2//)
1003 FORMAT(13A6,A2)
2000 FORMAT(1H0,5X'POINTS',15,3X'XMIN,XMAX,YMIN,YMAX'=' ,4F10.3)
2001 FORMAT(1H0,'BAD POINT',14,2(3X A3),4F10.3)
2002 FORMAT(1H ,14,3X A3,3X A3,F10.3,F9.3,F13.3,F9.3)
2003 FORMAT(1H0,'LINE LEFT RIGHT XL YL XR
      *YR'//)
2004 FORMAT(1H+,11X' ')
2005 FORMAT(1H ,14A6)
2006 FORMAT(1H1,14A6/)
3001 FORMAT(13)
C
      DPR=57.2957P
      FACT=1.
      IF(IONOF .EQ. 'OFF') FACT=2.
      HT=.125/FACT
      CALL FACTOR ( FACT )
C
5 NPTS=0
      REWIND 25
      READ(5,1001,END=400) DATA
      WRITE(27) DATA
      WRITE(6,2006) DATA
      XMIN=1000.
      XMAX=-1000.
      YMIN=1000.
      YMAX=-1000.
1 CONTINUE

```

ORIGINAL PAGE IS  
OF POOR QUALITY

```

READ ( 5,1000, END=200,ERR=2 ) NUM1, NUM2, X1, Y1, X2, Y2
IF ( NUM1 .EQ. '000' ) GO TO 1
IF(NUM1 .EQ. ' ' .AND. NUM2 .EQ. ' ') GO TO 1
IF ( FLD (6,6,NUM1 ) .EQ. '*****+') GO TO 1
NPTS=NPTS+1
XMIN=MIN(XMIN,X1,X2)
XMAX=MAX(XMAX,X1,X2)
YMIN=MIN(YMIN,Y1,Y2)
YMAX=MAX(YMAX,Y1,Y2)
WRITE(25) NUM1,NUM2,X1,Y1,X2,Y2
GO TO 1
2 READ(0,1003) DATA
WRITE(6,2005) DATA
GO TO 1
200 CONTINUE
IF ( NPTS .EQ. 0 ) GO TO 400
END FILE 25
DO 500 I12=1,2
NL=0
IF(I12 .EQ. 2) GO TO 202
YSIZE=YMAX-YMIN
WRITE(6,2000) NPTS,XMIN,XMAX,YMIN,YMAX
WRITE(6,2003)
202 CONTINUE
CALL SYMBOL(0.,-.25,.140,DATA,0.,60)
REWIND 25
201 CONTINUE
READ(25,END=300) NUM1,NUM2,X1,Y1,X2,Y2
NL=NL+1
IF(I12 .EQ. 1) WRITE(6,2002) NL,NUM1,NUM2,X1,Y1,X2,Y2
IF(I12 .EQ. 2) ENCODE(3001,NL) NL
CALL SYMBOL(X1-XMIN,Y1-YMIN,.06,1,0.,-1)
CALL SYMBOL(X2-XMIN,Y2-YMIN,.06,2,0.,-2)
AL=SQRT ((X2-X1)**2+(Y2-Y1)**2)
IF ( AL .GE. 3*HT ) GO TO 203
IF(I12 .EQ. 1) WRITE(6,2004)
GO TO 201
203 XC=(X1+X2)/2.
YC=(Y1+Y2)/2.
CALL ANGLE(X2-X1,Y2-Y1,THETA,$205)
THD=THETA*OPR
CS=COS(THETA)
SN=SIN(THETA)
XOF=1.5*HT*CS
YOF=1.5*HT*SN
XOF1=.020*SN
YOF1=.020*CS
IF(I12 .EQ. 2) NUM1=NL1
CALL SYMBOL(XC-XOF-XOF1-XMIN,YC-YOF+YOF1-YMIN,HT,NUM1,THD,3)
IF(I12 .EQ. 2) GO TO 201
XOF2=(.020+HT)*SN
YOF2=(.020+HT)*CS
CALL SYMBOL(XC-XOF+XOF2-XMIN,YC-YOF-YOF2-YMIN,HT,NUM2,THD,3)
GO TO 201
205 WRITE(6,2001) NPTS,NUM1,NUM2,X1,Y1,X2,Y2
GO TO 201
300 CONTINUE
IF(YSIZE*FACT .LT. 16.) GO TO 305
CALL PLOT(XMAX-XMIN+2.,0.,-3)
GO TO 500
305 IF(I12 .EQ. 1) CALL PLOT(0.,YSIZE+1.,-3)

```

```

      IF(I12 .EQ. 2) CALL PLOT(XMAX-XMIN+2.,-YSIZE-1.,-3)
500 CONTINUE
      GO TO 5
400 CONTINUE
      RETURN
      END
*TFOR, ISFBT   ANGLE,ANGLE
      SUBROUTINE ANGLE(X,Y,THETA,S)
C
C      SUBROUTINE TO DETERMINE ANGLE (0 TO 2*PI) FROM X-Y PAIR
C
      PI=3.1415926
C
      IF(X)10,20,30
C
C      SECOND OR THIRD QUADRANT
C
10 IF(Y)11,12,11
11 THETA=PI+ATAN(Y/X)           " THIRD QUADRANT
      RETURN
12 THETA=PI                     " 180 DEGREES
      RETURN
C
C      ON Y AXIS (PI/2 OR 3*PI/2)
C
20 IF(Y)21,22,23
21 THETA=1.5*PI                " 270 DEGREES
      RETURN
22 RETURN 4                     " DEGENERATE CASE
23 THETA=PI/2.                 " 90 DEGREES
      RETURN
C
C      FIRST OR FOURTH QUADRANT
C
30 IF(Y)31,32,31
31 THETA=ATAN(Y/X)             "
      IF(THETA .LT. 0.) THETA=2.*PI+THETA
      RETURN
32 THETA=0.
      RETURN
      END
*TFOR, ISFBT   BNDRY,BNDRY
      SUBROUTINE BNDRY
      DIMENSION DATA(2)
C
C      1000 FORMAT(5X A3,7X A3,2X 4F6.3)
C      1001 FORMAT(6X A6,A1//)
C      2000 FORMAT(1H ,A3,4F10.3)
C
C
      DATA HT/,125/
      DATA XMIN,XMAX,YMIN,YMAX/1000.,-1000.,1000.,-1000./
      NPTS=0
      REWIND 25
      REWIND 26
      CALL FACTOR(1.)
5 CONTINUE
      PEAD(5,1001,END=400) DATA
      XMN=1000.
      XMX=-1000.
      YMN=1000.

```

```

      YMX=-1000.
1  CONTINUE
  READ(5,1000,END=200) NUM1,NUM2,X1,Y1,X2,Y2
  IF(NUM2 .NE. ' ') GO TO 1
  IF(NUM1 .EQ. ' ') GO TO 1
  NPTS=NPTS+1
  XMIN=MIN(XMIN,X1,X2)
  XMAX=MAX(XMAX,X1,X2)
  YMIN=MIN(YMIN,Y1,Y2)
  YMAX=MAX(YMAX,Y1,Y2)
  XMN=MIN(XMN,X1,X2)
  XMX=MAX(XMX,X1,X2)
  YMN=MIN(YMN,Y1,Y2)
  YMX=MAX(YMX,Y1,Y2)
  WRITE(25) NUM1,X1,Y1,X2,Y2
  GO TO 1
200 CONTINUE
  WRITE(26) DATA,XMN,XMX,YMN,YMX
  GO TO 5
400 CONTINUE
  IF(NPTS .EQ. 0) RETURN
  END FILE 25
  REWIND 25
  REWIND 26
401 CONTINUE
  READ(26,END=410) DATA,XMN,XMX,YMN,YMX
  CALL SYMBOL((XMN+XMX)/2.-XMIN-3.5*HT,(YMN+YMX)/2.-YMIN-.5*HT,HT,
    * DATA,0,0,7)
  GO TO 401
410 CONTINUE
411 READ(25,END=420) NUM1,X1,Y1,X2,Y2
C  WRITE(6,2000) NUM1,X1,Y1,X2,Y2
  CALL SYMBOL(X1-XMIN,Y1-YMIN,.06,1,0.,-1)
  CALL SYMBOL(X2-XMIN,Y2-YMIN,.06,2,0.,-2)
  GO TO 410
420 CONTINUE
  CALL PLOT(XMAX-XMIN+4.,0.,-3)
  RETURN
  END

```

ORIGINAL PAGE IS  
OF POOR QUALITY



APPENDIX IV  
ROTATE PROGRAM LISTING

C	YOU MUST DIMENSION THE NUMBER OF ROWS OF A(*,4), THE NUMBER OF COLUMNS	00010
C	OF AT(4,*) AND ATAT(4,*), AND THE NUMBER OF COMPONENTS OF B(*) AND	00020
C	C(*) EQUAL TO TWO TIMES THE NUMBER OF CONTROL POINTS USED ON MAP	00030
	DOUBLE PRECISION A(12,4), AT(4,12), ATAT(4,12), B(12), C(12),	00040
	IX(12), Y(12), XD(12), YD(12), U(4), ATA(4,4)	00050
	DIMENSION K1(4), L1(4), THETA(2), THECOM(2,3), IX(12), IY(12)	00060
	INTEGER XX	00070
10	FORMAT(15)	00080
11	FORMAT(8F9.1)	00090
12	FORMAT(2I5,3F10.3)	00100
13	FORMAT(12F6.3)	00110
213	FORMAT(2X,12F7.3)	00120
14	FORMAT(1H1)	00130
15	FORMAT(//////, ' FOLLOWING IS A CHECK ON THE MAP ROTATION, ORIGIN DI	00140
	SPACEMENT, AND SCALING FACTOR')	00150
16	FORMAT(//////, ' THE TWO INDEPENDENT SOLUTIONS FOR THE CALIFORNIA ZO	00160
	NE 7 STATE PLANE COORDINATES OF THE')	00170
229	FORMAT(' DIGITIZATION ORIGIN (X0,Y0) AND THE TABLE TO MAP ROTATION	00180
	1 ANGLE THETA ')	00190
17	FORMAT(//,14X,'X0',19X,'Y0',14X,'THETA')	00200
18	FORMAT(/,2F20.1,3X,F4.0,' DEG ',F4.0,' MIN ',F4.0,' SEC')	00210
19	FORMAT(2I5,A3,I4,I2)	00220
219	FORMAT(1X,2I5,A3,I4,I2)	00230
20	FORMAT(4F6.3,I5)	00240
220	FORMAT(' THE SCALING FACTOR S = ',F10.1)	00250
221	FORMAT(//, ' MAP NUMBER, NUMBER OF POLYGONS FOR MAP, LAND USE OR PO	00260
	LYGON CODE, POLYGON TYPE')	00270
222	FORMAT(//, ' FOR THIS POLYGON-MIN X, MIN Y, MAX X, MAX Y, NUMBER OF	00280
	1 POINTS IN POLYGON')	00290
223	FORMAT(//, ' ALL DIGITIZED COORDINATES AND TRANSFORMED ZONE 7')	00300
224	FORMAT(' CALIFORNIA STATE PLANE COORDINATES FOR THIS POLYGON')	00310
232	FORMAT(' FOLLOW IN GROUPS OF TWELVE PAIRS')	00320
233	FORMAT(//)	00330
225	FORMAT(//, ' YOUR X0 ORIGIN REFERENCE SOLUTIONS DISAGREE BY MORE TH	00340
	1AN ',F10.3,' INCHES FOR MAPSETUP ',I5,' FOR CONTROL POINTS ',8I3)	00350
226	FORMAT(//, ' YOUR Y0 ORIGIN REFERENCE SOLUTIONS DISAGREE BY MORE TH	00360
	1AN ',F10.3,' INCHES FOR MAPSETUP ',I5,' FOR CONTROL POINTS ',8I3)	00370
227	FORMAT(//, ' YOUR THETA ROTATION SOLUTIONS DISAGREE BY MORE THAN ',	00380
	IF10.3,' MINUTES FOR MAP SETUP ',I5)	00390
228	FORMAT(4I10,I5)	00400
329	FORMAT(4I9,I5)	00410
230	FORMAT(' FOR THESE CONTROL POINTS, THE MAP POSITION IS OFF BY ')	00420
231	FORMAT(F15.3)	00430
235	FORMAT(8I10)	00440
236	FORMAT(8I9)	00450
C	*****	T0451
C	* PHASE 1 *	T0452
C	* CALCULATE NORTH-SOUTH ROTATION ANGLE, SCALING FACTOR & *	T0453
C	* TEST FOR VARIATION IN CONTROL PARAMETERS > THAN TOLERANCE *	T0454
C	*****	T0455
	READ(5,10) KSET	00460
C		T0461
C	READ KSET - NUMBER OF MAP SETUPS	T0462
C		T0463
C	BEGIN MAJOR LOOP OF PROGRAM - IBIG	T0464
C		T0465
	DO 100 IBIG = 1,KSET	00470
	READ(5,12) NPOLY,M,XOTOL, YOTOL, THETOL	00480
C		T0481
C	READ POLYGON TOLERANCE CONTROL CARD	T0482
C		T0483

ORIGINAL PAGE IS  
OF POOR QUALITY

C	NPOLY	TOTAL NUMBER OF POLYGONS	T0484
C	M	NUMBER OF CONTROL POINTS	T0485
C	XOTOL	X COORDINATE TOLERANCE LEVEL	T0486
C	YOTOL	Y COORDINATE TOLERANCE LEVEL	T0487
C	THETOL	ROTATION ANGLE TOLERANCE LIMIT	T0488
C			T0489
C			T0490
	WRITE(6,12)NPOLY,M,XOTOL,YOTOL,THETOL		00491
	M3 = MOD(M,4)		00500
	IF(M3.NE.0) GO TO 23		00510
	M3 = 4		00520
23	IF(M.LE.4) GO TO 21		00530
	M1 = (M-1)/4		00540
	DO 25 I2 = 1,M1		00550
	ISUB = 4*(I2-1)		00560
25	READ(5,11) (X(ISUB+13),Y(ISUB+13),I3 = 1,4)		00570
C			T0571
C	READ CALCULATED SPC CONTROL POINTS		T0572
C			T0573
C	X CONTROL POINT X		T0574
C	Y CONTROL POINT Y		T0575
C			T0576
	ISUB = 4*M1		00580
	GO TO 22		00590
21	ISUB = 0		00600
22	READ(5,11) (X(ISUB + 13),Y(ISUB + 13), I3 = 1,M3)		00610
	M3 = MOD(M,6)		00620
	IF(M3.NE.0) GO TO 28		00630
	M3 = 6		00640
28	IF(M.LE.6) GO TO 27		00650
	M1 = (M-1)/6		00660
	DO 30 I2 = 1,M1		00670
	ISUB = 6*(I2-1)		00680
30	READ(5,13) (XD(ISUB+13), YD(ISUB+13), I3 = 1,6)		00690
C			T0691
C	READ DIGITIZED CONTROL POINTS		T0692
C			T0693
C	XD DIGITIZED CONTROL POINT X		T0694
C	YD DIGITIZED CONTROL POINT Y		T0695
C			T0696
	ISUB = 6*M1		00700
	GO TO 29		00710
27	ISUB = 0		00720
29	READ(5,13) (XD(ISUB+13),YD(ISUB+13),I3 = 1,M3)		00730
C			T0731
C	INITIAL ARRAYS AND INPUT DATA		T0732
C	FROM CALCULATED & DIGITIZED COORDINATES		T0733
C			T0734
	MM1 = 2*M		00740
	DO 50 I = 1,M		00750
	I1 = 2*I-1		00760
	A(I1,1) = 1.		00770
	A(I1,2) = 0.		00780
	A(I1,3) = XD(I)		00790
	A(I1,4) = YD(I)		00800
	B(I) = X(I)		00810
	I1 = I1+1		00820
	A(I1,1) = 0.		00830
	A(I1,2) = 1.		00840
	A(I1,3) = YD(I)		00850
	A(I1,4) = -XD(I)		00860

ORIGINAL PAGE IS  
OF POOR QUALITY

50	B(I1) = Y(I)	00870
	DO 34 I = 1,4	00880
	DO 34 J = 1,MM1	00890
34	AT(I,J) = A(J,I)	00900
	CALL RATMUL(ATA,AT,A,4,MM1,4)	00910
	CALL INVERS(ATA,4)	00920
	CALL RATMUL(ATAT,ATA,AT,4,4,MM1)	00930
	DO 36 I = 1,4	00940
	U(I) = 0.	00950
	DO 36 J = 1,MM1	00960
36	U(I) = U(I) + ATAT(I,J)*B(J)	00970
60	UX = U(3)*U(3) + U(4)*U(4)	00980
	S = SQRT(UX)	00990
	UX=U(3)/S	01000
	THETA(1) = ARCCOS(UX)	01010
	UX=U(4)/S	01020
	THETA(2) = ARSIN(UX)	01030
	IF(U(3).GE.0.) GO TO 263	01040
	THETA(1) = 3.14159265-THETA(1)	01050
263	CONTINUE	01060
	DO 45 I = 1,2	01070
	THETA(I) = ABS(THETA(I))	01080
	IF(U(3).GE.0.) GO TO 260	01090
	IF(U(4).GE.0.) GO TO 261	01100
	THETA(I) = 3.14159265 + THETA(I)	01110
	GO TO 262	01120
261	THETA(I) = 3.14159265 - THETA(I)	01130
	GO TO 262	01140
260	IF(U(4).GE.0.) GO TO 262	01150
	THETA(I) = 6.28318530 - THETA(I)	01160
262	CONTINUE	01170
	THETA(I) = THETA(I)*180./3.14159265	01180
	THECOM(I,1) = AINT(THETA(I))	01190
	DUM = (THETA(I)-THECOM(I,1))*60.	01200
	THECOM(I,2) = AINT(DUM)	01210
	DUM = (DUM-THECOM(I,2))*60.	01220
45	THECOM(I,3) = AINT(DUM)	01230
C		T1231
C	WRITE HEADING "FOLLOWING IS A CHECK ON"	T1232
C		T1233
	WRITE(6,14)	01240
	WRITE(6,15)	01250
	WRITE(6,16)	01260
	WRITE(6,229)	01270
	WRITE(6,17)	01280
	DO 47 I = 1,2	01290
47	WRITE(6,18) U(1),U(2),THECOM(I,1),THECOM(I,2),THECOM(I,3)	01300
C		T1301
C	WRITE RESULTS OF ROTATION FROM CALCULATED AND DIGITIZED POINTS	T1302
C		T1303
C	U(1) ORIGIN X0	T1304
C	U(2) ORIGIN Y0	T1305
C	THECOM(I,1) ROTATION DEGREES	T1306
C	THECOM(I,2) ROTATION MINUTES	T1307
C	THECOM(I,3) ROTATION SECONDS	T1308
C		T1309
	WRITE(6,220) S	01310
C		T1311
C	WRITE SCALING FACTOR - S	T1312
C		T1313
C	TEST FOR ROTATION ANGLE AND SCALING FACTOR WITHIN TOLERANCE	T1314

C	DO 32 I = 1,MM1	T1315
	C(I) = 0.	01320
	DO 32 J = 1,4	01330
32	C(I) = C(I) + A(I,J)*U(J)	01340
	DO 33 I = 1,MM1	01350
	UX=(C(I)-B(I))/S	01360
	C(I) = ABS(UX)	01370
33	CONTINUE	01380
	K = 0	01390
	L = 0	01400
	DO 37 I = 1,M	01410
	I1 = 2*I-1	01420
	IF(C(I1).LT.XOTOL) GO TO 40	01430
	K = K+1	01440
	K1(K) = I	01450
40	I1 = 2*I	01460
	IF(C(I1).LT.YOTOL) GO TO 37	01470
	L = L+1	01480
	L1(L) = I	01490
37	CONTINUE	01500
	IF(K.NE.0) GO TO 97	01510
	IF(L.NE.0) GO TO 97	01520
	IF(ABS((THETA(1)-THETA(2))*60.).GT.THETOL) GO TO 97	01530
C		01540
C	COMPLETION OF TOLERANCE CHECKS	T1541
C		T1542
C	*****	T1543
C	* PHASE I1 *	T1544
C	* SCALE AND ROTATE DIGITIZED POINTS, MIN-MAX & BOUNDARIES *	T1545
C	* CODED WITHIN THE FRAMWORK OF THE GEO-REF SYSTEM - PHASE I *	T1546
C	*****	T1547
C		T1548
C		T1549
65	DO 65 I = 1,2	01550
	THETA(I) = THETA(I)*3.14159265/180.	01560
	THETA(1) = (THETA(1)+THETA(2))/2.	01570
	A(1,1) = S*COS(THETA(1))	01580
	A(1,2) = S*SIN(THETA(1))	01590
	A(2,1) = -A(1,2)	01600
	A(2,2) = A(1,1)	01610
	WRITE(6,14)	01620
	DO 102 IB = 1,NPOLY	01630
	READ(5,19) MAPNO,MAPOL,LUCODE,PCLYNO,PTYPE	01640
C		T1641
C	READ POLYGON CONTROL CARD	T1642
C		T1643
C	MAPNO MAP NUMBER FOR THE POLYGONS THAT FOLLOW	T1644
C	MAPOL TOTAL NUMBER OF POLYGONS ON MAP	T1645
C	LUCODE LAND USE OF POLYGON CODE	T1646
C	POLYNO SEQUENCE NUMBER OF POLYGON U TO MAPOL	T1647
C	PTYPE POLYGON TYPE-KEY TO DONUT POLYGONS	T1648
C		T1649
C	NORMAL POLYGONS ARE CODED 10	T1650
		T1651
C	READ(5,20) XMIN,YMIN,XMAX,YMAX,NP	T1652
C	READ MINMUM & MAXIMUM COORDINATES	T1653
C		T1654
C	XMIN MINIMUM X COORDINATE	T1655
C	YMIN MINIMUM Y COORDINATE	T1656
C	XMAX MAXIMUM X COORDINATE	T1657
C	YMAX MAXIMUM Y COORDINATE	T1658

C	NP	NUMBER OF POINTS IN POLYGON	T1659
C			T1660
		XX = (NP-1)/6	01661
		NP1 = 1 + XX	01670
		ISE = NP-6*(NP1-1)	01680
		NPF = (NP1+1)/2	01690
		NPER = (2*NPE)-NP1	01700
		WRITE(6,14)	01710
		WRITE(6,221)	01720
		WRITE(6,219)MAPNO,MAPOL,LUCODE,POLYNO,PTYPE	01730
C			T1731
C		WRITE POLYGON CONTROL CARD	T1732
C			T1733
		WRITE(7,19) MAPNO,MAPOL,LUCODE,POLYNO,PTYPE	01740
		WRITE(6,222)	01750
		DO 75 I = 1,4	01760
		IF(I.EQ.4) GO TO 79	01770
		IF(I.EQ.3) GO TO 78	01780
		IF(I.EQ.2) GO TO 77	01790
C			T1791
C		ROTATE AND SCALE MINIMUM AND MAXIMUM COORDINATES	T1792
C			T1793
		DUM1 = XMIN	01800
		DUM2 = YMIN	01810
		GO TO 76	01820
77		DUM1 = XMAX	01830
		DUM2 = YMIN	01840
		GO TO 76	01850
78		DUM1 = XMIN	01860
		DUM2 = YMAX	01870
		GO TO 76	01880
79		DUM1 = XMAX	01890
		DUM2 = YMAX	01900
76		C(I) = U(I) + A(1,1)*DUM1 + A(1,2)*DUM2	01910
75		B(I) = U(2) + A(2,1)*DUM1 + A(2,2)*DUM2	01920
		XMIN = C(1)	01930
		DO 80 I = 2,4	01940
		IF(XMIN.LE.C(I)) GO TO 80	01950
		XMIN = C(I)	01960
80		CONTINUE	01970
		XMAX = C(1)	01980
		DO 82 I = 2,4	01990
		IF(XMAX.GE.C(I)) GO TO 82	02000
		XMAX = C(I)	02010
82		CONTINUE	02020
		YMIN = B(1)	02030
		DO 84 I = 2,4	02040
		IF(YMIN.LE.B(I)) GO TO 84	02050
		YMIN = B(I)	02060
84		CONTINUE	02070
		YMAX = B(1)	02080
		DO 86 I = 2,4	02090
		IF(YMAX.GE.B(I)) GO TO 86	02100
		YMAX = B(I)	02110
86		CONTINUE	02120
		IXMIN = INT(XMIN*10)	02130
		IYMIN = INT(YMIN*10)	02140
		IXMAX = INT(XMAX*10)	02150
		IYMAX = INT(YMAX*10)	02160
C			T2161
C		ROTATION AND SCALING COMPLETED FOR MIN-MAX COORDINATES	T2162

C		T2163
	WRITE(6,228) IXMIN,IYMIN,IXMAX,IYMAX,NP	02170
C		T2171
C	WRITE MINIMUM AND MAXIMUM COORDINATES TO PRINTER AND CARD FILE	T2172
C		T2173
	WRITE(7,329) IXMIN,IYMIN,IXMAX,IYMAX,NP	02180
	WRITE(6,223)	02190
	WRITE(6,224)	02200
	WRITE(6,232)	02210
C		T2211
C	BEGIN SCALING AND ROTATION FOR POLYGON BOUNDARY	T2212
C		T2213
	DO 55 ISM = 1,NPE	02220
	WRITE(6,233)	02230
C		T2231
C	COMPLEX ROUTINE FOR COMPENSATING TO VARIATION IN INPUT FORMAT	T2232
C		T2233
C	NP NUMBER OF COORDINATE PAIRS	T2234
C	NPU NUMBER OF CARDS TO BE READ	T2235
C	I3E PARTIAL COORDINATE PAIRS REMAINING AFTER COMPLETE CARD READ	T2236
C	NPE NUMBER OF GROUPS OF COORDINATES	T2237
C	NPER 1-IF LAST SET CONTAINS ONLY ONE CARD	T2238
C		T2239
	IF(ISM.NE.NPE) GO TO 61	02240
	IF(NPER.EQ.1) GO TO 63	02250
	READ(5,13) (XD(I2),YD(I2), I2 = 1,6)	02260
	IBEG = 7	02270
	IEND = 6 + I3E	02280
	GO TO 64	02290
63	IREG = 1	02300
	IEND = I3E	02310
C		T2311
C	READ POLYGON POINT CARDS IN SETS OF TWO	T2312
C		T2313
64	READ(5,13) (XD(I2), YD(I2), I2= IBEG,IEND)	02320
	GO TO 67	02330
61	CONTINUE	02340
	DO 62 IL=1,2	02350
	LSUB = 6*(IL-1)	02360
62	READ(5,13) (XD(LSUB+I2),YD(LSUB+I2),I2=1,6)	02370
67	CONTINUE	02380
	IF(ISM.EQ.NPE) GO TO 68	02390
	NPP = 12	02400
	GO TO 69	02410
68	NPP = 6*(1-NPER) + I3E	02420
69	CONTINUE	02430
C		T2431
C	END OF POLYGON READ ROUTINE	T2432
C		T2433
C	CALCULATE ROTATION SCALING FACTOR	T2434
C		T2435
	DO 70 I = 1,NPP	02440
	X(I) = U(1) + A(1,1)*XD(I) + A(1,2)*YD(I)	02450
70	Y(I) = U(2) + A(2,1)*XD(I) + A(2,2)*YD(I)	02460
	IF(ISM.EQ.NPE) GO TO 81	02470
	DO 140 IL=1,2	02480
	LSUB = 6*(IL-1)	02490
140	WRITE(6,213) (XD(LSUB+I2),YD(LSUB+I2),I2=1,6)	02500
C		T2501
C	WRITE COORDINATES TO PRINTER	T2502
C		T2503

GO TO 72	02510
81 IF(NPER.EQ.1) GO TO 141	02520
WRITE(6,213) (XD(I2),YD(I2), I2=1,6)	02530
IBEG = 7	02540
IEND = 6+I3E	02550
GO TO 142	02560
141 IBEG = 1	02570
IEND = I3E	02580
142 WRITE(6,213) (XD(I2),YD(I2), I2=IBEG,IEND)	02590
72 IF(ISM.EQ.NPE) GO TO 74	02600
C	T2601
C CONVER COORDINATES TO INTEGER	T2602
C	T2603
DO 56 I2 = 1,I2	02610
IX(I2) = X(I2)*10	02620
56 IY(I2) = Y(I2)*10	02630
DO 143 IL=1,3	02640
LSUB = 4*(IL-1)	02650
WRITE(6,235) (IX(LSUB+I2),IY(LSUB+I2),I2=1,4)	02660
C	T2661
C WRITE INTEGER COORDINATES TO CARD FILE	T2662
C	T2663
143 WRITE(7,236) (IX(LSUB+I2),IY(LSUB+I2),I2=1,4)	02670
GO TO 55	02680
74 XX = (NPP-1)/4	02690
NP2 = 1+XX	02700
I4E = NPP - 4*(NP2-1)	02710
DO 57 I2=1,NPP	02720
IX(I2) = X(I2)*10	02730
57 IY(I2) = Y(I2)*10	02740
DO 145 IL = 1,NP2	02750
LSUB = 4*(IL-1)	02760
IF(IL.EQ.NP2) GO TO 147	02770
WRITE(6,235) (IX(LSUB+I2),IY(LSUB+I2), I2=1,4)	02780
WRITE(7,236) (IX(LSUB+I2),IY(LSUB+I2), I2=1,4)	02790
GO TO 145	02800
147 WRITE(6,235) (IX(LSUB+I2),IY(LSUB+I2), I2=1,I4E)	02810
WRITE(7,236) (IX(LSUB+I2),IY(LSUB+I2), I2=1,I4E)	02820
145 CONTINUE	02830
55 CONTINUE	02840
102 CONTINUE	02850
GO TO 100	02860
C	T2861
C ERROR CONDITION	T2862
C	T2863
97 IF(K.EQ.0) GO TO 98	02870
WRITE(6,225) XOTOL,IBIG,(K1(I), I = 1,K)	02880
WRITE(6,230)	02890
DO 87 I = 1,K	02900
IWR = 2*K1(I)-1	02910
87 WRITE(6,231) C(IWR)	02920
98 IF(L.EQ.0) GO TO 99	02930
WRITE(6,226) YOTOL,IBIG,(L1(I), I = 1,L)	02940
WRITE(6,230)	02950
DO 88 I = 1,L	02960
IWR = 2*L1(I)	02970
88 WRITE(6,231) C(IWR)	02980
99 IF(ABS((THETA(1)-THETA(2))*60.).LE.THETOL) GO TO 100	02990
WRITE(6,227) THETOL,IBIG	03000
100 CONTINUE	03010
STOP	03020



END	03030
SUBROUTINE RATMUL(C,A,B,I,J,K)	03040
DOUBLE PRECISION C(4,4), A(I,J), B(J,K)	03050
DO 100 IR = 1,I	03060
DO 100 JC = 1,K	03070
C(IR,JC) = 0.	03080
DO 100 L = 1,J	03090
C(IR,JC) = C(IR,JC) + A(IR,L)*B(L,JC)	03100
100 CONTINUE	03110
RETURN	03120
END	03130
SUBROUTINE INVERS(ATA,N)	03140
DOUBLE PRECISION ATA(N,N), A(4,4)	03150
DIMENSION XTA(4,4)	03160
302 FORMAT(1X, ' NO SOLUTION. A 0 EXISTS ON THE MAIN DIAGONAL FOR')	03170
303 FORMAT(' ROW NUMBER ',I3,' COLUMN NUMBER ',I3)	03180
301 FORMAT(' REMEMBER-ROWS MAY HAVE BEEN SWITCHED PREVIOUS TO THIS ')	03190
304 FORMAT(' POINT. NO MORE SWITCHING POSSIBLE')	03200
DO 300I = 1,N	03210
DO 306 JJ = 1,N	03220
XTA(I,JJ)=ATA(I,JJ)	03230
306 A(I,JJ) = 0.0	03240
300 A(I,I) = 1.0	03250
DET = 1.0	03260
DO 341 IH = 1,N	03270
IG = IH	03280
IF (ABS(XTA(IH,IH)).GE.1.E-10) GO TO 332	03290
C SWAP ROWS	03300
305 IG = IG+1	03310
IF (IG.LE.N) GO TO 308	03320
WRITE(6,302)	03330
WRITE(6,303) IH, IH	03340
WRITE(6,301)	03350
WRITE(6,304)	03360
GO TO 381	03370
308 IF (ABS(XTA(IG,IG)).LT.1.E-10) GO TO 305	03380
DET = (-1.0)*DET	03390
DO 324JJ = 1,N	03400
C = ATA(IH,JJ)	03410
D = A(IH,JJ)	03420
ATA(IH,JJ) = ATA(IG,JJ)	03430
A(IH,JJ) = A(IG,JJ)	03440
ATA(IG,JJ) = C	03450
324 A(IG,JJ) = D	03460
332 DIAG = ATA(IH,IH)	03470
DET = DET*DIAG	03480
DO 334JJ = 1,N	03490
ATA(IH,JJ) = ATA(IH,JJ)/DIAG	03500
334 A(IH,JJ) = A(IH,JJ)/DIAG	03510
IF(IH.EQ.N) GO TO 336	03520
II = IH+1	03530
DO 340I = II, N	03540
DIAG = ATA(I,IH)	03550
DO 340JJ = 1,N	03560
ATA(I,JJ) = ATA(I,JJ) - (DIAG*ATA(IH,JJ))	03570
340 A(I,JJ) = A(I,JJ) - (DIAG*A(IH,JJ))	03580
341 CONTINUE	03590
336 IN1 = N-1	03600
DO 348 I = 1,IN1	03610
IH = N+1-I	03620
II = IH - 1	03630

ORIGINAL PAGE IS  
OF POOR QUALITY

352	DIAG = ATA(II,IH)	03640
	IG = N	03650
346	IF(IG .EQ. 0) GO TO 342	03660
	ATA(II,IG) = ATA(II,IG) - (DIAG*ATA(IH,IG))	03670
	A(II,IG) = A(II,IG) - (DIAG*A(IH,IG))	03680
	IG = IG - 1	03690
	GO TO 346	03700
342	IF(II .EQ. 1) GO TO 348	03710
	II = II-1	03720
	GO TO 352	03730
348	CONTINUE	03740
354	DO 380 I = 1,N	03750
	DO 380 J = 1,N	03760
380	ATA(I,J) = A(I,J)	03770
381	RETURN	03780
	END	03790

APPENDIX V  
CHAIN PROGRAM LISTING



C	BEGIN MAJOR LOOP OF PROGRAM - IBIG	T0534
C		T0535
	DO 100 IBIG = 1,KSET	00540
	READ(5,12) NPOLY,M,XOTOL,YOTOL,THETOL	00550
C		T0551
C	READ POLYGON TOLERANCE CONTROL CARD	T0552
C		T0553
C	NPOLY TOTAL NUMBER OF POLYGONS	T0554
C	M NUMBER OF CONTROL POINTS	T0555
C	XOTOL X COORDINATE TOLERANCE LEVEL	T0556
C	YOTOL Y COORDINATE TOLERANCE LEVEL	T0557
C	THETOL ROTATION ANGLE TOLERANCE LIMIT	T0558
C		T0559
C	COMPLEX ROUTINE TO READ COORDINATES IN STRING FORMAT	T0560
C		T0561
	M3 = MOD(M,4)	00562
	IF(M3.NE.0) GO TO 23	00570
	M3 = 4	00580
23	IF(M.LE.4) GO TO 21	00590
	M1 = (M-1)/4	00600
	DO 25 I2 = 1,M1	00610
	ISUB = 4*(I2-1)	00620
25	READ(5,11) (X(ISUB+I3),Y(ISUB+I3),I3 = 1,4)	00630
C		T0631
C	READ CALCULATED SPC CONTROL POINTS	T0632
C		T0633
C	X CONTROL POINT X	T0634
C	Y CONTROL POINT Y	T0635
C		T0636
	ISUB = 4*M1	00640
	GO TO 22	00650
21	ISUB = 0	00660
22	READ(5,11) (X(ISUB + I3),Y(ISUB + I3), I3 = 1,M3)	00670
	M3 = MOD(M,6)	00680
	IF(M3.NE.0) GO TO 28	00690
	M3 = 6	00700
28	IF(M.LE.6) GO TO 27	00710
	M1 = (M-1)/6	00720
	DO 30 I2 = 1,M1	00730
	ISUB = 6*(I2-1)	00740
30	READ(5,13) (XD(ISUB+I3), YD(ISUB+I3), I3 = 1,6)	00750
C		T0751
C	READ DIGITIZED CONTROL POINTS	T0752
C		T0753
C	XD DIGITIZED CONTROL POINT X	T0754
C	YD DIGITIZED CONTROL POINT Y	T0755
C		T0756
	ISUB = 6*M1	00760
	GO TO 29	00770
27	ISUB = 0	00780
29	READ(5,13) (XD(ISUB+I3),YD(ISUB+I3),I3 = 1,M3)	00790
C		T0791
C	INITIAL ARRAYS AND INPUT DATA	T0792
C	FROM CALCULATED & DIGITIZED COORDINATES	T0793
C		T0794
	MM1 = 2*M	00800
	DO 50 I = 1,M	00810
	I1 = 2*I-1	00820
	A(I1,1) = 1.	00830
	A(I1,2) = 0.	00840
	A(I1,3) = XD(I)	00850

ORIGINAL PAGE IS  
OF POOR QUALITY

	A(I1,4) = YD(I)	00860
	B(I1) = X(I)	00870
	I1 = I1+1	00880
	A(I1,1) = 0.	00890
	A(I1,2) = 1.	00900
	A(I1,3) = YD(I)	00910
	A(I1,4) = -XD(I)	00920
50	B(I1) = Y(I)	00930
	DO 34 I = 1,4	00940
	DO 34 J = 1,MM1	00950
34	AT(I,J) = A(J,I)	00960
	CALL RATMUL(ATA,AT,A,4,MM1,4)	00970
	CALL INVERS(ATA,4)	00980
	CALL RATMUL(ATAT,ATA,AT,4,4,MM1)	00990
	DO 36 I = 1,4	01000
	U(I) = 0.	01010
	DO 36 J = 1,MM1	01020
36	U(I) = U(I) + ATAT(I,J)*B(J)	01030
60	S = SQRT(U(3)*U(3) + U(4)*U(4))	01040
	THETA(1) = ARCOS(U(3)/S)	01050
	THETA(2) = ARSIN(U(4)/S)	01060
	IF(U(3).GE.0.) GO TO 263	01070
	THETA(1) = 3.14159265-THETA(1)	01080
263	CONTINUE	01090
	DO 45 I = 1,2	01100
	THETA(I) = ABS(THETA(I))	01110
	IF(U(3).GE.0.) GO TO 260	01120
	IF(U(4).GE.0.) GO TO 261	01130
	THETA(I) = 3.14159265 + THETA(I)	01140
	GO TO 262	01150
261	THETA(I) = 3.14159265 - THETA(I)	01160
	GO TO 262	01170
260	IF(U(4).GE.0.) GO TO 262	01180
	THETA(I) = 6.28318530 - THETA(I)	01190
262	CONTINUE	01200
	THETA(I) = THETA(I)*180./3.14159265	01210
	THECOM(I,1) = AINT(THETA(I))	01220
	DUM = (THETA(I)-THECOM(I,1))*60.	01230
	THECOM(I,2) = AINT(DUM)	01240
	DUM = (DUM-THECOM(I,2))*60.	01250
45	THECOM(I,3) = AINT(DUM)	01260
C		T1261
C	WRITE HEADING "FOLLOWING IS A CHECK ON"	T1262
C		T1263
C		T1264
C	WRITE RESULTS OF ROTATION FROM CALCULATED AND DIGITIZED POINTS	T1265
C		T1266
C	U(1) ORIGIN X0	T1267
C	U(2) ORIGIN Y0	T1268
C	THECOM(I,1) ROTATION DEGREES	T1269
C	THECOM(I,2) ROTATION MINUTES	T1270
C	THECOM(I,3) ROTATION SECONDS	T1271
C		T1272
	WRITE(6,14)	01273
	WRITE(6,15)	01280
	WRITE(6,16)	01290
	WRITE(6,229)	01300
	WRITE(6,17)	01310
	DO 47 I = 1,2	01320
47	WRITE(6,18) U(1),U(2),THECOM(I,1),THECOM(I,2),THECOM(I,3)	01330
	WRITE(6,220) S	01340

C		T1341
C	WRITE SCALING FACTOR - S	T1342
C		T1343
C	TEST FOR ROTATION ANGLE AND SCALING FACTOR WITHIN TOLERANCE	T1344
C		T1345
	DO 32 I = 1,MM1	01350
	C(I) = 0.	01360
	DO 32 J = 1,4	01370
32	C(I) = C(I) + A(I,J)*U(J)	01380
	DO 33 I = 1,MM1	01390
33	C(I) = ABS((C(I)-B(I))/S)	01400
	K = 0	01410
	L = 0	01420
	DO 37 I = 1,M	01430
	I1 = 2*I-1	01440
	IF(C(I1).LT.X0TOL) GO TO 40	01450
	K = K+1	01460
	K1(K) = I	01470
40	I1 = 2*I	01480
	IF(C(I1).LT.Y0TOL) GO TO 37	01490
	L = L+1	01500
	L1(L) = I	01510
37	CONTINUE	01520
	IF(K.NE.0) GO TO 97	01530
	IF(L.NE.0) GO TO 97	01540
	IF(ABS((THETA(1)-THETA(2))*60.).GT.THETOL) GO TO 97	01550
C		T1551
C	COMPLETION OF TOLERANCE CHECKS	T1552
C		T1553
	DO 65 I = 1,2	01560
65	THETA(I) = THETA(I)*3.14159265/180.	01570
	THETA(1) = (THETA(1)+THETA(2))/2.	01580
	A(1,1) = S*COS(THETA(1))	01590
	A(1,2) = S*SIN(THETA(1))	01600
	A(2,1) = -A(1,2)	01610
	A(2,2) = A(1,1)	01620
	WRITE(6,14)	01630
C		T1631
C	*****	T1632
C	* PHASE II *	T1633
C	* REVERSE AND DUPLICATE SEGMENTS WHICH SERVE AS A BOUNDARY *	T1634
C	* TO TWO BLOCKS AND CHECK FOR SAME BEGINNING AND ENDING COORD *	T1635
C	*****	T1636
C		T1637
	DO 102 IB = 1,NPOLY	01640
102	READ(5,245) ILEFT(IB),ITRITE(IB),XL(IB),YL(IB),XR(IB),YR(IB)	01650
	NP1 = 2*NPOLY - 1	01660
	DO 370 I = 1,NPOLY	01670
	ILEFT(NPOLY+I) = ITRITE(I)	01680
	ITRITE(NPOLY+I) = ILEFT(I)	01690
	XL(NPOLY + I) = XR(I)	01700
	YL(NPOLY+I) = YR(I)	01710
	XR(NPOLY+I) = XL(I)	01720
370	YR(NPOLY+I) = YL(I)	01730
C	WE WANT TO GET RID OF ALL TRACT BOUNDARY SEGMENTS WITH OTHER THAN	01740
C	PRESENT TRACT ON LEFT.	01750
C	WE ASSUME THAT LAST SEGMENT DIGITIZED (NPOLY) HAS OUR DESIRED TRACT	01760
C	ON LEFT. IT IS AN INTERNAL BLOCK.	01770
	NURTRK = ILEFT(NPOLY)/10000	01780
	I = 0	01790
372	I = I+1	01800

ORIGINAL PAGE 13  
OF POOR QUALITY

363	IACHTR = ILEFT(I)/10000	01810
	IF(I.GT.NP1) GO TO 375	01820
	IF(NURTRK.EQ.IACHTR) GO TO 372	01830
	ON 373 I1 = I,NP1	01840
	ILEFT(I1) = ILEFT(I1+1)	01850
	ITRITE(I1) = IRITE(I1+1)	01860
	XL(I1) = XL(I1+1)	01870
	YL(I1) = YL(I1+1)	01880
	XR(I1) = XR(I1+1)	01890
373	YR(I1) = YR(I1+1)	01900
	NP1 = NP1-1	01910
	GO TO 363	01920
375	IF(NURTRK.EQ.IACHTR) GO TO 377	01930
	NP1 = NP1 - 1	01940
377	CONTINUE	01950
	CALL SORT6(ILEFT,ITRITE,XL,YL,XR,YR,NP1)	01960
	NP1 = NP1+1	01970
	I = 1	01980
	IBLKS = 0	01990
374	IBLKS = IBLKS + 1	02000
	IBLOCK(IBLKS) = ILEFT(I)	02010
	KTBL(IBLKS) = 0	02020
376	KTBL(IBLKS) = KTBL(IBLKS) + 1	02030
	I = I + 1	02040
	IF(I.GT.NP1) GO TO 378	02050
	IF((IBLOCK(IBLKS)).NE.(ILEFT(I))) GO TO 374	02060
	GO TO 376	02070
378	CONTINUE	02080
C IBLKS IS THE NUMBER OF BLOCKS IN THIS TRACT		02090
C NOW WE'LL CHAIN THE BLOCK TOGETHER IN SEQUENTIAL ORDER		02100
	IBEGF = 0	02110
	IENDF = 0	02120
	DO 380 I10 = 1,IBLKS	02130
	NSEG = KTBL(I10)	02140
	NUMBLK = IBLOCK(I10)	02150
	IBEGF = IENDF + 1	02160
	IENDF = IENDF + NSEG	02170
	NCULS = 0	02180
	IMARK = IENDF	02190
	DO 382 I1 = IBEGF, IENDF	02200
	IF(IMARK.LT.I1) GO TO 386	02210
	IF(NUMBLK.NE.ITRITE(I1)) GO TO 382	02220
385	IF(NUMBLK.EQ.ITRITE(IMARK)) GO TO 384	02230
C SWITCH CUL DE SAC TO END OF BLOCK SEGMENTS		02240
	IDUM2 = IRITE(I1)	02250
	DUM3 = XL(I1)	02260
	DUM4 = YL(I1)	02270
	DUM5 = XR(I1)	02280
	DUM6 = YR(I1)	02290
	ITRITE(I1) = IRITE(IMARK)	02300
	XL(I1) = XL(IMARK)	02310
	YL(I1) = YL(IMARK)	02320
	XR(I1) = XR(IMARK)	02330
	YR(I1) = YR(IMARK)	02340
	ITRITE(IMARK) = IDUM2	02350
	XL(IMARK) = DUM3	02360
	YL(IMARK) = DUM4	02370
	XR(IMARK) = DUM5	02380
	YR(IMARK) = DUM6	02390
C FINISHED SWITCHING		02400
	GO TO 382	02410



384	NCULS = NCULS + 1	02420
	IMARK = IMARK - 1	02430
	IF(IMARK .LT. 11) GO TO 386	02440
	GO TO 385	02450
382	CONTINUE	02460
386	CONTINUE	02470
C	CUL DE SACS ARE NOW AT END OF BLOCK SEGMENTS	02480
	NSEG1 = NSEG - NCULS	02490
C	NOW WE'LL MATCH ENDS OF SEGMENTS TO BEGINNINGS OF OTHER SEGMENTS	02500
C	NSEG1 IS THE NUMBER OF BLOCK SEGMENTS EXCLUDING CUL DE SACS	02510
	I1 = IBEGF	02520
	IEND1 = IENDF - NCULS	02530
	IS = IBEGF	02540
390	IS = IS + 1	02550
393	IF(IS .GT. IEND1) GO TO 392	02560
	IF(ABS(XR(I1) - XL(IS)) .GT. XOTOL) GO TO 390	02570
	IF(ABS(YR(I1) - YL(IS)) .GT. YOTOL) GO TO 390	02580
C	IS MARKS THE NEXT SEGMENT LINK OF THIS BLOCK	02590
C	SWITCH THIS SEGMENT BACK TO THE ONE FOLLOWING I1	02600
	I1 = I1 + 1	02610
	IDUM2 = ITRITE(I1)	02620
	DUM3 = XL(I1)	02630
	DUM4 = YL(I1)	02640
	DUM5 = XR(I1)	02650
	DUM6 = YR(I1)	02660
	ITRITE(I1) = ITRITE(IS)	02670
	XL(I1) = XL(IS)	02680
	YL(I1) = YL(IS)	02690
	XR(I1) = XR(IS)	02700
	YR(I1) = YR(IS)	02710
	ITRITE(IS) = IDUM2	02720
	XL(IS) = DUM3	02730
	YL(IS) = DUM4	02740
	XR(IS) = DUM5	02750
	YR(IS) = DUM6	02760
	IS = I1 + 1	02770
	GO TO 393	02780
392	IF(I1 .EQ. IEND1) GO TO 394	02790
C	BLOCK DID NOT CHAIN IF YOU REACHED HERE	02800
395	WRITE(6,241) NUMBLK	02810
	DO 396 I2 = IBEGF, IENDF	02820
396	WRITE(6,240) ITRITE(I2), ITRITE(I2), XL(I2), YL(I2), XR(I2), YR(I2)	02830
	GO TO 380	02840
394	IF(ABS(XR(IEND1) - XL(IBEGF)) .GT. XOTOL) GO TO 395	02850
	IF(ABS(YR(IEND1) - YL(IBEGF)) .GT. YOTOL) GO TO 395	02860
C	IF YOU REACHED HERE, BLOCK CHAINED OK	02870
C	NOW WE'LL AVERAGE THE DOUBLY DIGITIZED NODES	02880
C	SINCE THE DIGITIZATION AGREES WITHIN TOLERANCE LIMITS	02890
C		T2891
C	*****	T2892
C	* PHASE III *	T2893
C	* SCALE AND ROTATE DIGITIZED POINTS, MIN-MAX & BOUNDARIES *	T2894
C	* CODED WITHIN THE FRAMEWORK OF THE GEO-REF SYSTEM - PHASE I *	T2895
C	*****	T2896
C		T2897
	MAPNO = NUMBLK/10000	02900
	MAPOL = NUMBLK - (MAPNO*10000)	02910
	LUCODE = MAPOL/10	02920
	NPOLYD = 110	02930
	NPTYP = 10	02940
	XH(1) = (XL(IBEGF) + XR(IEND1))/2.	02950

	YH(1) = (YL(1BEGF)+YR(1END1))/2.	02960
	NP = NSEGI+1	02970
	XH(NP) = XH(1)	02980
	YH(NP) = YH(1)	02990
	DO 300 I1 = 2,NSEGI	03000
	IMARK1 = 1BEGF+I1-2	03010
	IMARK2 = 1BEGF+I1-1	03020
	XH(I1) = (XR(IMARK1)+XL(IMARK2))/2.	03030
300	YH(I1) = (YR(IMARK1)+YL(IMARK2))/2.	03040
316	CONTINUE	03050
	XMIN = XH(1)	03060
	YMIN = YH(1)	03070
	XMAX = XH(1)	03080
	YMAX = YH(1)	03090
	DO 302 I1 = 2,NP	03100
	IF(XMIN .LE. XH(I1)) GO TO 302	03110
	XMIN = XH(I1)	03120
302	CONTINUE	03130
	DO 304 I1 = 2,NP	03140
	IF(YMIN .LE. YH(I1)) GO TO 304	03150
	YMIN = YH(I1)	03160
304	CONTINUE	03170
	DO 306 I1 = 2,NP	03180
	IF(XMAX .GE. XH(I1)) GO TO 306	03190
	XMAX = XH(I1)	03200
306	CONTINUE	03210
	DO 308 I1 = 2,NP	03220
	IF(YMAX .GE. YH(I1)) GO TO 308	03230
308	CONTINUE	03240
	YMAX = YH(I1)	03250
	XX = (NP-1)/6	03260
	NP1 = 1 +XX	03270
	I3E = NP-6*(NP1-1)	03280
	NPE = (NP1+1)/2	03290
	NPER = (2*NPE)-NP1	03300
	WRITE(6,14)	03310
	WRITE(6,221)	03320
	WRITE(6,219)MAPNO,MAPOL,LUCODE,NPOLYO,NPTYP	03330
	WRITE(7,19) MAPNO,MAPOL,LUCODE,NPOLYO,NPTYP	03340
	WRITE(6,222)	03350
	DO 75 I = 1,4	03360
	IF(I.EQ.4) GO TO 79	03370
	IF(I.EQ.3) GO TO 78	03380
	IF(I.EQ.2) GO TO 77	03390
	DUM1 = XMIN	03400
	DUM2 = YMIN	03410
	GO TO 76	03420
77	DUM1 = XMAX	03430
	DUM2 = YMIN	03440
	GO TO 76	03450
78	DUM1 = XMIN	03460
	DUM2 = YMAX	03470
	GO TO 76	03480
79	DUM1 = XMAX	03490
	DUM2 = YMAX	03500
76	C(I) = U(I) + A(1,1)*DUM1 + A(1,2)*DUM2	03510
75	B(I) = U(2) + A(2,1)*DUM1 + A(2,2)*DUM2	03520
	XMIN = C(1)	03530
	DO 80 I = 2,4	03540
	IF(XMIN.LE.C(I)) GO TO 80	03550
	XMIN = C(I)	03560

80	CONTINUE	03570
	XMAX = C(1)	03580
	DO 82 I = 2,4	03590
	IF(XMAX.GE.C(1)) GO TO 82	03600
	XMAX = C(1)	03610
82	CONTINUE	03620
	YMIN = B(1)	03630
	DO 84 I = 2,4	03640
	IF(YMIN.LE.B(1)) GO TO 84	03650
	YMIN = B(1)	03660
84	CONTINUE	03670
	YMAX = B(1)	03680
	DO 86 I = 2,4	03690
	IF(YMAX.GE.B(1)) GO TO 86	03700
	YMAX = B(1)	03710
86	CONTINUE	03720
	IXMIN = INT(XMIN*10)	03730
	IYMIN = INT(YMIN*10)	03740
	IXMAX = INT(XMAX*10)	03750
	IYMAX = INT(YMAX*10)	03760
	WRITE(6,228) IXMIN,IYMIN,IXMAX,IYMAX,NP	03770
	WRITE(7,228) IXMIN,IYMIN,IXMAX,IYMAX,NP	03780
	WRITE(6,223)	03790
	WRITE(6,224)	03800
	WRITE(6,232)	03810
	DO 55 ISM = 1,NPE	03820
	WRITE(6,233)	03830
	IF(ISM.NE.NPE) GO TO 61	03840
	IF1 = (NPE-1)*2*6	03850
	IF(NPER.EQ.1) GO TO 63	03860
	DO 310 I2 = 1,6	03870
	XD(I2) = XH(IF1+I2)	03880
310	YD(I2) = YH(IF1+I2)	03890
	IBFG = 7	03900
	IEND = 6 + I3E	03910
	GO TO 64	03920
63	IBEG = 1	03930
	IEND = I3E	03940
64	DO 312 I2 = IBEG,IEND	03950
	XD(I2) = XH(IF1+I2)	03960
312	YD(I2) = YH(IF1+I2)	03970
	GO TO 67	03980
61	CONTINUE	03990
	DO 314 I2 = 1,12	04000
	IF1 = (ISM-1)*2*6	04010
	XD(I2) = XH(IF1+I2)	04020
314	YD(I2) = YH(IF1+I2)	04030
67	CONTINUE	04040
	IF(ISM.EQ.NPE) GO TO 68	04050
	NPP = 12	04060
	GO TO 69	04070
68	NPP = 6*(1-NPER) + I3E	04080
69	CONTINUE	04090
	DO 70 I = 1,NPP	04100
	X(I) = U(1) + A(1,1)*XD(I) + A(1,2)*YD(I)	04110
70	Y(I) = U(2) + A(2,1)*XD(I) + A(2,2)*YD(I)	04120
	IF(ISM.EQ.NPE) GO TO 81	04130
	DO 140 IL=1,2	04140
	LSUB = 6*(IL-1)	04150
140	WRITE(6,213) (XD(LSUB+I2),YD(LSUB+I2),I2=1,6)	04160
	GO TO 72	04170

ORIGINAL PAGE IS  
OF POOR QUALITY

81	IF(NPER.EQ.1) GO TO 141	04180
	WRITE(6,213) (XD(I2),YD(I2), I2=1,6)	04190
	IBEG = 7	04200
	IEND = 6+I3E	04210
	GO TO 142	04220
141	IBEG = 1	04230
	IEND = I3E	04240
142	WRITE(6,213) (XD(I2),YD(I2), I2=IBEG,IEND)	04250
72	IF(ISM.EQ.NPE) GO TO 74	04260
	DO 56 I2 = 1,I2	04270
	IX(I2) = X(I2)*10	04280
56	IY(I2) = Y(I2)*10	04290
	DO 143 IL=1,3	04300
	LSUB = 4*(IL-1)	04310
	WRITE(6,235) (IX(LSUB+I2),IY(LSUB+I2),I2=1,4)	04320
143	WRITE(7,235) (IX(LSUB+I2),IY(LSUB+I2),I2=1,4)	04330
	GO TO 55	04340
74	XX = (NPP-1)/4	04350
	NP2 = 1+XX	04360
	I4E = NPP - 4*(NP2-1)	04370
	DO 57 I2=1,NPP	04380
	IX(I2) = X(I2)*10	04390
57	IY(I2) = Y(I2)*10	04400
	DO 145 IL = 1,NP2	04410
	LSUB = 4*(IL-1)	04420
	IF(IL.EQ.NP2) GO TO 147	04430
	WRITE(6,235) (IX(LSUB+I2),IY(LSUB+I2), I2=1,4)	04440
	WRITE(7,235) (IX(LSUB+I2),IY(LSUB+I2), I2=1,4)	04450
	GO TO 145	04460
147	WRITE(6,235) (IX(LSUB+I2),IY(LSUB+I2), I2=1,I4E)	04470
	WRITE(7,235) (IX(LSUB+I2),IY(LSUB+I2), I2=1,I4E)	04480
145	CONTINUE	04490
55	CONTINUE	04500
	IF(NCULS .EQ. 0) GO TO 320	04510
	NPTYP = 99	04520
	WRITE(6,243)	04530
	NP = NCULS*2	04540
	NPM1 = NP-1	04550
	NCULS = 0	04560
	DO 320 I1 = 1,NPM1,2	04570
	IMARK = (I1+2)/2	04580
	XH(I1) = XL(IEND1 + IMARK)	04590
	YH(I1) = YL(IEND1 + IMARK)	04600
	XH(I1+1) = XR(IEND1 + IMARK)	04610
320	YH(I1+1) = YR(IEND1 + IMARK)	04620
	GO TO 316	04630
380	CONTINUE	04640
	GO TO 100	04650
97	IF(K.EQ.0) GO TO 98	04660
	WRITE(6,225) XOTOL,IBIG,(KI(I), I = 1,K)	04670
	WRITE(6,230)	04680
	DO 87 I = 1,K	04690
	IWR = 2*KI(I)-1	04700
87	WRITE(6,231) C(IWR)	04710
98	IF(L.EQ.0) GO TO 99	04720
	WRITE(6,226) YOTOL,IBIG,(LI(I), I = 1,L)	04730
	WRITE(6,230)	04740
	DO 88 I = 1,L	04750
	IWR = 2*LI(I)	04760
88	WRITE(6,231) C(IWR)	04770
99	IF(ABS((THETA(1)-THETA(2))*60.)).LE.THETOL) GO TO 100	04780

	WRITE(6,227) THETOL,IBIG	04790
100	CONTINUE	04800
	STOP	04810
	END	04820
	SUBROUTINE SORT6(ITL,ITR,XL,YL,XR,YR,N)	04830
	DIMENSION XL(1000), YL(1000), XR(1000), YR(1000),ITL(1000),	04840
	ITR(1000)	04850
	DO 20 I = 1,N	04860
	IF(ITL(I) .LE. ITL(I+1)) GO TO 20	04870
	K = I + 2	04880
10	K = K-1	04890
	IF(K .LT. 2) GO TO 20	04900
	IF(ITL(K-1) .LE. ITL(K)) GO TO 20	04910
	IDUM1 = ITL(K)	04920
	IDUM2 = ITR(K)	04930
	DUM3 = XL(K)	04940
	DUM4 = YL(K)	04950
	DUM5 = XR(K)	04960
	DUM6 = YR(K)	04970
	ITL(K) = ITL(K-1)	04980
	ITR(K) = ITR(K-1)	04990
	XL(K) = XL(K-1)	05000
	YL(K) = YL(K-1)	05010
	XR(K) = XR(K-1)	05020
	YR(K) = YR(K-1)	05030
	ITL(K-1) = IDUM1	05040
	ITR(K-1) = IDUM2	05050
	XL(K-1) = DUM3	05060
	YL(K-1) = DUM4	05070
	XR(K-1) = DUM5	05080
	YR(K-1) = DUM6	05090
	GO TO 10	05100
20	CONTINUE	05110
	RETURN	05120
	END	05130
	SUBROUTINE RATMUL(C,A,B,I,J,K)	05140
	DIMENSION C(I,K), A(I,J), B(J,K)	05150
	DO 100 IR = 1,I	05160
	DO 100 JC = 1,K	05170
	C(IR,JC) = 0.	05180
	DO 100 L = 1,J	05190
	C(IR,JC) = C(IR,JC) + A(IR,L)*B(L,JC)	05200
100	CONTINUE	05210
	RETURN	05220
	END	05230
	SUBROUTINE INVERS(ATA,N)	05240
	DIMENSION ATA(N,N), A(4,4)	05250
302	FORMAT(IX, ' NO SLOUTION. A 0 EXISTS ON THE MAIN DIAGONAL FOR')	05260
303	FORMAT(' ROW NUMBER ',I3,' COLUMN NUMBER ',I3)	05270
301	FORMAT(' REMEMBER-ROWS MAY HAVE BEEN SWITCHED PREVIOUS TO THIS ')	05280
304	FORMAT(' POINT. NO MORE SWITCHING POSSIBLE')	05290
	DO 300I = 1,N	05300
	DO 306 JJ = 1,N	05310
306	A(I,JJ) = 0.0	05320
300	A(I,I) = 1.0	05330
	DET = 1.0	05340
	DO 341 IH = 1,N	05350
	IG = IH	05360
	IF (ABS(ATA(IH,IH)).GE.1.E-10) GO TO 332	05370
C SWAP ROWS		05380
305	IG = IG+1	05390

IF (IG.LE.N) GO TO 308	05400
WRITE(6,302)	05410
WRITE(6,303) IH, IH	05420
WRITE(6,301)	05430
WRITE(6,304)	05440
GO TO 381	05450
308 IF (ABS(ATA(IG,IG)).LT.1.E-10) GO TO 305	05460
DET = (-1.0)*DET	05470
DO 324JJ = 1,N	05480
C = ATA(IH,JJ)	05490
D = A(IH,JJ)	05500
ATA(IH,JJ) = ATA(IG,JJ)	05510
A(IH,JJ) = A(IG,JJ)	05520
ATA(IG,JJ) = C	05530
324 A(IG,JJ) = D	05540
332 DIAG = ATA(IH,IH)	05550
DET = DET*DIAG	05560
DO 334JJ = 1,N	05570
ATA(IH,JJ) = ATA(IH,JJ)/DIAG	05580
334 A(IH,JJ) = A(IH,JJ)/DIAG	05590
IF(IH.EQ.N) GO TO 336	05600
II = IH+1	05610
DO 340I = II, N	05620
DIAG = ATA(I,IH)	05630
DO 340JJ = 1,N	05640
ATA(I,JJ) = ATA(I,JJ) - (DIAG*ATA(IH,JJ))	05650
340 A(I,JJ) = A(I,JJ) - (DIAG*A(IH,JJ))	05660
341 CONTINUE	05670
336 IN1 = N-1	05680
DO 348 I = 1,IN1	05690
IH = N+1-I	05700
II = IH - 1	05710
352 DIAG = ATA(II,IH)	05720
IG = N	05730
346 IF(IG.EQ. 0) GO TO 342	05740
ATA(II,IG) = ATA(II,IG) - (DIAG*ATA(IH,IG))	05750
A(II,IG) = A(II,IG) - (DIAG*A(IH,IG))	05760
IG = IG - 1	05770
GO TO 346	05780
342 IF(II.EQ. 1) GO TO 348	05790
II = II-1	05800
GO TO 352	05810
348 CONTINUE	05820
354 DO 380 I = 1,N	05830
DO 380 J = 1,N	05840
380 ATA(I,J) = A(I,J)	05850
381 RETURN	05860
END	05870

APPENDIX VI  
DACS PROGRAM LISTING





5 RCODE = LCODE + 1	DAC00600
IF( CODE .EQ. BLANK ) GO TO 9	DAC00610
SELECT = 3	DAC00620
IF( CODE .EQ. SLCT ) SELECT = 100	DAC00630
IF( CODE .EQ. RJCT ) SELECT = 0	DAC00640
IF( SELECT .EQ. 3 ) GO TO 4	DAC00650
8 ECODE = 100	DAC00660
IF( ELECT .EQ. BLOC ) ECODE = 7	DAC00670
IF( ELECT .EQ. BGPS ) ECODE = 7	DAC00680
IF( ELECT .EQ. TRAC ) ECODE = 11	DAC00690
IF( ELECT .EQ. REGN ) ECODE = 9	DAC00700
IF( ECODE .GT. 11 ) GO TO 4	DAC00730
C	DAC00720
9 AMULT = 0	DAC00740
IF( AREA .EQ. ACRES ) AMULT = 640	DAC00750
IF( AREA .EQ. SQFT ) AMULT = 5280 * 5280	DAC00760
IF( AREA .EQ. SOMI ) AMULT = 1	DAC00770
IF( AMULT .EQ. 0 ) GO TO 4	DAC00710
C	DAC00780
WRITE( OUTFIL, 1090 ) AMULT, AREA, CALC, OPTION, TITLE	DAC00790
1090 FORMAT( 18X, I11, 2A4, I4, 3A4 )	DAC00800
IF( CODE .EQ. BLANK ) GO TO 80	DAC00810
10 READ( 5, 22, END=80 ) TRACT	DAC00820
C	DAC00830
C THE VARIABLE 'TRACT' MAY ACTUALLY BE BLOCK, BG, OR REGION	DAC00840
C	DAC00850
22 FORMAT( 19 )	DAC00860
C	DAC00870
IF( TRACT .GE. NINES ) GO TO 80	DAC00880
C	DAC00890
IF (ELECT.EQ.BLOC) TRACT = TRACT*1000 + BLOCK	DAC00900
IF(COUNT) 30, 30, 40	DAC00910
30 SLIST(1)=TRACT	DAC00920
COUNT=1	DAC00930
GO TO 10	DAC00940
40 DO 50 I=1, COUNT	DAC00950
IF(SLIST(I) - TRACT) 50, 10, 60	DAC00960
50 CONTINUE	DAC00970
COUNT=COUNT + 1	DAC00980
SLIST(COUNT)=TRACT	DAC00990
GO TO 10	DAC01000
60 COUNT=COUNT + 1	DAC01010
J=COUNT	DAC01020
70 K=J - 1	DAC01030
SLIST(J)=SLIST(K)	DAC01040
J=J - 1	DAC01050
IF(J .GT. 1) GO TO 70	DAC01060
SLIST(1)=TRACT	DAC01070
GO TO 10	DAC01080
80 IF( COUNT .EQ. 0 ) WRITE( 6,1040)	DAC01090
C	DAC01100
IF( COUNT .GT. 0 ) WRITE( 6,1050) COUNT, ELECT, CODE	DAC01110
X (SLIST(N),N=1,COUNT)	DAC01120
C	DAC01130
1040 FORMAT( '0 ALL AREAS TO BE PROCESSED' )	DAC01140
1050 FORMAT( '0 LIST OF', 16, 2X, A4, ' TO BE ', A4, 'CTED.' //	DAC01150
X ( / 12I'0 ) )	DAC01160
C	DAC01170
500 RECIN=0	DAC01180
RECOUT=0	DAC01190
C	DAC01200

ORIGINAL PAGE IS  
OF POOR QUALITY

C		DAC01210
C	READ DIME FILE RECORD AND SELECT REQUIRED INFORMATION	
C		
	550 IF(FILE.EQ.BNRY) GO TO 555	
	READ(INFILE,560,END=800)RECD(1),RECD(3),RECD(5),LINK,RECD(10),	
	XRECD(11),RECD(2),RECD(4),RECD(8),RECD(9),RECD(6),	
	XRECD(7),FX,FY,TX,TY	
	560 FORMAT(28X,A1,16X,2(13,2X),29X,17,216,19X,414,2(13X,15,1X),	
	X54X,417)	
	GO TO 558	
	555 READ(INFILE,END=800) IREC	
	558 CONTINUE	
C		
C	NONE STREET CODE	
	RECD(1)=IREC(22)	
C	FROM NODE NUMBER AND MAP NUMBER	
	RECD(2)=IREC(3)	
	RECD(3)=IREC(4)	
C	TO NODE NUMBER AND MAP NUMBER	
	RECD(4)=IREC(6)	
	RECD(5)=IREC(7)	
C	BLOCK NUMBER, LEFT AND RIGHT	
	RECD(6)=IREC(30)	
	RECD(7)=IREC(31)	
C	PLACE CODE, LEFT AND RIGHT	
	RECD(8)=IREC(44)	
	RECD(9)=IREC(45)	
C	TRACT NUMBER, LEFT AND RIGHT	
	RECD(10)=IREC(32)	
	RECD(11)=IREC(33)	
C	STATE PLANE COORDINATES FOR FROM AND TO NODES	
	FX=XREC(70)	
	FY=XREC(71)	
	TX=XREC(73)	
	TY=XREC(74)	
C		
C	SAVE RECORD GEO-CODES	
	DO 90 J=1,11	
	RECS(J)=RECD(J)	
	90 CONTINUE	
C		
C		
C	SEQUENCE NUMBER	
C		
	LINK=IREC(13)	
C		
	RECIN=RECIN + 1	DAC01290
C		DAC01300
C		DAC01310
C	MAKE BLOCK NUMBERS UNIQUE BY COMBINING WITH TRACT NUMBERS.	DAC01320
C		DAC01330
	IF( LCODE .NE. 6        ) GO TO 5651	DAC01340
C		DAC01350
	DO 565 K = 6,7	DAC01360
	J = K+4	DAC01370
	RECD(K) = MOD(RECD(K),10) + RECD(K)/100*10	
	RECD(J) = MOD(RECD(J),10) + RECD(J)/100*10	
	RECD( K ) = RECD(K) + RECD(J)*10000	DAC01380
	IF( CALC .EQ. BGPS ) RECD(K) = RECD(K) / 100	DAC01390
	565 CONTINUE	DAC01400
C		DAC01410

C		DAC01420
C		DAC01430
C	SELECT ONLY BOUNDARY SEGMENTS.	DAC01440
C		DAC01450
C		DAC01460
	5651 KEYL = RECD( LCODE )	DAC01470
	KEYR = RECD( RCODE )	DAC01480
	IF (KEYL.EQ.KEYR) GO TO 550	DAC01490
C		DAC01500
C		DAC01510
C		
C	ENCODE NODE AND MAP NUMBERS, ALL RESULTING #'S UNIQUE IN TRACT	
	DO 566 I=2,4,2	
	RECD(I)=RECD(I)+10000*RECD(I+1)	
	566 CONTINUE	
C		
	IF (COUNT.EQ.0) GO TO 575	DAC01640
	IF( ISKIP( RECS( ECODE ) , SELECT ) .EQ. 1 ) GO TO 5801	
C		DAC01660
C		DAC01670
C	KEYR IS REGION TO LEFT OF SEGMENT WHEN DIRECTION IS REVERSED. TX, T	DAC01680
C	FROM NODE COORDINATES.	DAC01690
C		DAC01700
C		DAC01710
	575 WRITE (OUTFIL,570) KEYR, KEYL, LINK, RECD(4), RECD(2), TX,TY	DAC01720
	RECOUT=RECOUT + 1	DAC01730
	580 IF (COUNT.EQ.0) GO TO 581	DAC01740
	5801 IF( ISKIP( RECS( ECODE-1 ),SELECT ) .EQ. 1 ) GO TO 550	
C		DAC01760
C		DAC01770
C	KEYL IS COUNTERPART SEGMENT OF KEYR, TO DESCRIBE BOUNDARY OF ADJOIND	DAC01780
C	REGION.	DAC01790
C		DAC01800
C		DAC01810
	581 WRITE (OUTFIL,570) KEYL, KEYR, LINK, RECD(2), RECD(4), FX, FY	
	570 FORMAT( 3I9, 4I10)	
	RECOUT = RECOUT+1	DAC01830
	GO TO 550	DAC01840
	800 WRITE (OUTFIL,810) NINES	DAC01850
	810 FORMAT (I9)	DAC01870
	END FILE OUTFIL	DAC01880
	WRITE(6,850) RECN, RECOUT	DAC01890
	850 FORMAT('0 PRESORT PROCESSING COMPLETED' /// 10X, I6, ' RECORDS IN	DAC01900
	X' // 10X, I6, ' RECORDS PASSED TO SORT')	DAC01910
	CALL EXIT	DAC01920
	STOP	DAC01930
	END	DAC01940
	FUNCTION ISKIP ( KEY, SELECT )	DAC01950
	IMPLICIT INTEGER(A-Z)	DAC01960
	COMMON NCODE,COUNT,RECD,SLIST	DAC01970
	DIMENSION RECD(11),SLIST(2000)	DAC01980
	ISKIP = 0	DAC01990
	TOP=1	DAC02000
	BOTTOM=COUNT	DAC02010
	TRACT=KEY	DAC02020
	10 IF( BOTTOM .LT. TOP ) GO TO 301	DAC02030
	MIDPT = TOP + (BOTTOM-TOP)/2	DAC02040
	IF( TRACT - SLIST(MIDPT) ) 15, 100, 20	DAC02050
	15 BOTTOM = MIDPT - 1	DAC02060
	GO TO 10	DAC02070
	20 TOP = MIDPT + 1	DAC02080

ORIGINAL PAGE IS  
OF POOR QUALITY

GO TO 10  
100 ISKIP = 1  
IF( SELECT .EQ. 100)ISKIP=0  
RETURN  
301 IF( SELECT .EQ. 100) ISKIP = 1  
RETURN  
END

DAC02090  
DAC02100  
DAC02110  
DAC02120  
DAC02130  
DAC02140  
DAC02150

IMPLICIT INTEGER (8-Y)	DACC00010
INTEGER*2 ND(4)	DACC00020
DIMENSION NO(2),NS(3)	DACC00030
EQUIVALENCE (NO(1),ND(1))	DACC00040
DOUBLE PRECISION ZMULT	DACC00050
DIMENSION TITLE(3), OPTION(3)	DACC00060
DIMENSION TLIST(2000)	DACC00070
COMMON ELIST( 2000, 5 ) , ZMULT	DACC00080
	DACC00090
C DATA R, INFILE, OUT1, OUT2, EOF, MAX, AREA, ATOTAL, ASUBDACC00100	
X / 1, 9, 10, 11, 99999999, 2000, 0.0, 0.0, 0.0	DACC00110
LOGICAL PIOS / .FALSE. /	DACC00120
LOGICAL ADJ / .FALSE. /	DACC00130
LOGICAL LIST / .FALSE. / , BNDS / .FALSE. /	DACC00140
DATA NUM / 0 / , BGRP / 'BGRPS' /	DACC00150
DATA BLKS / 'BLKS' / , TDRPT / 0 / , DROPT / 0 /	DACC00160
DATA BLANK / ' ' / , IA / 'A' /	DACC00170
	DACC00180
C * OPTION PROCESSING	DACC00190
C	DACC00200
C	DACC00210
C	DACC00220
CNT=0	DACC00230
READ(5,5001,ERR=9999) AMULT,UNITS,KALC,OPTION,TITLE	DACC00240
5001 FORMAT(F10.0,2A4,3I1,3A4)	DACC00250
C	DACC00260
ZMULT = AMULT * .00000001	DACC00270
IF (OPTION(1) .EQ. 1) LIST = .TRUE.	DACC00280
IF (OPTION(2) .EQ. 1) BNDS = .TRUE.	DACC00290
IF (OPTION(3) .EQ. 1) ADJ = .TRUE.	DACC00300
READ( INFILE, 5002, END=500) KEY, TLIST(1), (ELIST(1,J),J=1,5)	DACC00310
5002 FORMAT(7I9)	DACC00320
WRITE(6,5014) TITLE, KALC, UNITS, OPTION, KEY	DACC00330
5014 FORMAT( '1', 10X, 'U. S. BUREAU OF CENSUS - DIME AREA-CENTROID	DACC00340
XSYSTEM (DACS)' // 25X, 3A4 // 11X, 'CALCULATIONS ARE FOR 'A4,	DACC00350
X ' IN 'A4,5X,3I1,' IS OPTION'/// 5X, 'FIRST AREA IS'.I11 // )	DACC00360
100 R=R+1	DACC00370
IF( R .LT. MAX ) GO TO 120	DACC00380
WRITE(6,5003) KALC, KEY, MAX	DACC00390
5003 FORMAT( // ' *** 'A4, I11, ' HAS',I5, ' CR MORE SEGMENTS' )	DACC00400
DROPT = DROPT + 1	DACC00410
NIX = -1	DACC00420
105 READ( INFILE, 5002, END=500)KEYX, TLIST(1), (ELIST(1,J),J=1,5)	DACC00430
NIX = NIX + 1	DACC00440
IF( KEYX .EQ. KEY ) GO TO 105	DACC00450
WRITE( 6, 5004 ) NIX	DACC00460
5004 FORMAT( 24X, I5, ' SEGMENTS CROPPED. NO CALCULATIONS' // )	DACC00470
NIX = 0	DACC00480
R = 2	DACC00490
C	DACC00500
C	DACC00510
120 READ( INFILE, 5002, END=500)KEYX, TLIST(R), (ELIST(R,J),J=1,5)	DACC00520
C	DACC00530
IF( KEYX .EQ. KEY ) GO TO 100	DACC00540
C	DACC00550
C \$ *	DACC00560
C \$ PROCESSING *	DACC00570
C \$ *	DACC00580
C	DACC00590
130 NUM = NUM + 1	DACC00600
C	DACC00610
NBLS = R-1	
AREA = 0.0	

CENTX = 0	DACC00620
CENTY = 0	DACC00630
IND = 0	DACC00640
CALL CHAIN ( NBL5, COMPS, CLOSES, RVSL5 )	DACC00650
IF( COMPS .NE. CLOSES ) GO TO 140	DACC00660
IF( COMPS .EQ. 1 ) GO TO 150	DACC00670
C	DACC00680
WRITE( 6,5005) KALC, KEY, COMPS	DACC00690
C	DACC00700
5005 FORMAT( '0*** ',A4, I11, ' HAS', I3, ' BOUNDED REGIONS - CHECK' )	DACC00710
GO TO 150	DACC00720
C	DACC00730
140 WRITE( 6,5006) KALC, KEY, COMPS, CLOSES	DACC00740
5006 FORMAT( '0*** ',A4, I11, ' HAS', I3, ' COMPONENTS. ONLY',I3, ' ARDAC 00750	DACC00760
X CLOSED. REGION NOT PROCESSED' )	DACC00770
C	DACC00780
WRITE( 6, 5007 )	DACC00790
DO 145 I = 1,NEL5	DACC00800
NQ(1) = ELIST(1,2)	DACC00810
NQ(2) = ELIST(1,3)	DACC00820
NS(1) = ELIST(1,1)	DACC00830
NS(2) = ELIST(1,4)	DACC00840
NS(3) = ELIST(1,5)	DACC00850
145 WRITE(6,5097) ND, NS	DACC00860
5097 FORMAT(2(5X,A2,I5),3I12)	DACC00870
C	DACC00880
5007 FORMAT( '0 SEGMENT LISTING: ' //	DACC00890
X FROM NODE TO NODE REC NUMBER FROM X FROM Y' //)	DACC00900
C	DACC00910
DROPT = DROPT + 1	DACC00920
C	DACC00930
GO TO 1000	DACC00940
C	DACC00950
150 CALL CALC( NBL5, AREA, CENTX, CENTY, NX, MY )	DACC00960
C	DACC00970
IF( AREA .LE. 0.0 ) DROPT = DROPT + 1	DACC00980
C	DACC00990
IF( CENTX .NE. 0 ) CALL POLYPT( NBL5, CENTX, CENTY, IND,NX,MY)	DACC01000
IF(.NOT. BNDS ) GO TO 1000	DACC01010
C	DACC01020
SYMAP=IA	DACC01030
KEY1=KEY/10000	DACC01040
KEY2=MOD(KEY,10000)	DACC01050
KEY2=KEY2*100	DACC01060
DO 180 I = 1, NBL5	DACC01070
KEY2=KEY2+I	DACC01080
WRITE(OUT2,5011) KEY1,SYMAP,ELIST(1,5),ELIST(1,4),ELIST(1,1),	DACC01090
XKEY2	DACC01100
5011 FORMAT(15,4X,A1,2I10,20X,I10,13X,I7)	DACC01110
SYMAP=BLANK	DACC01120
180 CONTINUE	DACC01130
C	DACC01140
KEY2=KEY2+1	DACC01150
WRITE(OUT2,5011) KEY1,SYMAP,ELIST(1,5),ELIST(1,4),ELIST(1,1),	DACC01160
XKEY2	DACC01170
C	DACC01180
1000 IF( LIST ) WRITE(6,5008)KALC, KEY, NBL5, AREA, UNITS, CENTX,CENTY	DACC01190
5008 FORMAT( //5X, A4, I11, I9, ' SEGMENTS. AREA =' , F15.5,1X, A4	DACC01200
X / 29X , ' CENTROID IS :',2I15 )	DACC01210
IF (PI05) CALL REFNT (NBL5,KEY)	DACC01220
C	

IF(RVSL5 .GT. 0) WRITE(6,5015) RVSL5	DAC 01230
5015 FORMAT( 9X, 111, ' REVERSALS - CHECK FOR PCSS. ERRORS.' )	DACC01240
IF( IND .GE. 0 ) GO TO 1100	DACC01250
C	DACC01260
CALL ADJUST ( NBL5, CENTX, CENTY, NODE )	DACC01270
C	DACC01280
WRITE(6,5009) NODE , CENTX, CENTY	DAC 01290
5009 FORMAT( / 9X, 'CENTROID WAS OUTSIDE BOUNDARY - ADJUSTED TO NODE'	DACC01300
X 111 / 29X , 'NEW CENTROID IS :',2115 )	DACC01310
C	DACC01320
1100 WRITE( OUT1, 5010 ) KEY, AREA, CENTX, CENTY	DACC01330
C	DACC01340
5010 FORMAT( 110, F20.5, 2110 )	DACC01350
C	DACC01360
ASUB = ASUB + AREA	DACC01390
ATOTAL= ATOTAL + AREA	DACC01400
C	DACC01410
1150 IF( ADJ ) CALL ADJNCY ( NELS, KEY , TLIST )	DACC01420
DO 1200 I=1,5	DAC 01430
1200 ELIST(1,I)= ELIST(R,I)	DACC01440
C	DACC01450
1250 KEY = KEYX	DACC01460
R = 2	DACC01470
C	DACC01480
C	DACC01490
1300 IF( KEY .LT. ECF ) GO TO 120	DACC01500
C	DACC01510
IF( TDRPT .EQ. 0 ) TDRPT = DRCPY	DACC01520
WRITE(6,5013) NUM, KALC, TDRPT, ATOTAL, UNITS	DACC01530
5013 FORMAT( '1' / 1X, 10('----') // 110, 1X, A4, ' , ' , 16, ' OMITTED	DACC01540
X // 5X, 'TOTAL AREA IS' , E13.5, 1X, A4 // 1X, 10('----') )	DACC01550
GO TO 1350	DAC 01560
500 KEYX= EOF	DACC01570
GO TO 1250	DACC01580
C	DACC01590
9999 WRITE(6,5099)	DACC01600
C	DACC01610
5099 FORMAT( '1 ERROR IN FIRST RECORD' )	DACC01620
WRITE(6,5001)1, AMULT, UNITS, KALC, OPTION, TITLE	DACC01630
1350 STOP	DAC 01640
END	DACC01650
SUBROUTINE CHAIN (NBL5,COMPS,CLOSES,RVSL5)	DAC 01660
C	DACC01670
C	DACC01680
C	DACC01690
C	DACC01700
C	DACC01710
C	DACC01720
C	DACC01730
C	DACC01740
IMPLICIT INTEGER (A-Z)	DACC01750
DOUBLE PRECISION ZMULT	DACC01760
DIMENSION HOLDER(5),ELIST(2000,5)	DACC01770
COMMON ELIST,ZMULT	DACC01780
BEGIN = 1	DACC01790
END = 1	DACC01800
COMPS = 0	DACC01810
RVSL5 = 0	DACC01820
CLOSES = 0	DACC01830
HEAD = ELIST(BEGIN,2)	DACC01840
TAIL = ELIST(END,3)	DACC01850
IF( NBL5 - 1 ) 1250, 1200, 1000	

1000	IF( HEAD .EQ. TAIL ) GO TO 1200	DACC01860
	START = END + 1	DACC01870
	IF( START .GT. NBLS ) GO TO 1200	DACC01880
	DO 1100 I = START, NBLS	DACC01890
	IF (ELIST(I,3).EQ.HEAD) GO TO 3000	DACC01900
	IF (ELIST(I,2).EQ.TAIL) GO TO 2000	DACC01910
1100	CONTINUE	DACC01920
	DO 1150 I = START, NBLS	DACC01930
	II = I + 1	DACC01940
	IF (I.EQ.NBLS) II = 1	DACC01950
	IF (ELIST(II,2).EQ.HEAD) GO TO 2990	DACC01960
	IF (ELIST(II,3).EQ.TAIL) GO TO 1990	DACC01970
1150	CONTINUE	DACC01980
1200	CMPS = CMPS + 1	DACC01990
	IF( HEAD .EQ. TAIL ) CLOSSES = CLOSSES + 1	DACC02000
1250	IF( END .GE. NELS ) RETURN	DACC02010
CHAIN **		DACC02020
	ELIST(END,1) = -ELIST(END,1)	DACC02030
	IF (CLOSSES.GE.2) GO TO 1400	DACC02040
C		DACC02050
C	THE FOLLOWING SECTION REARRANGES THE ORDER OF SEGMENTS IN THE CHAIN	DACC02060
C	PERMIT CHAINING AS ONE COMPONENT FOR FIGURE EIGHT OR CHECKERBOARD	DACC02070
C	CONFIGURATIONS. VIZ.	DACC02080
C		DACC02090
C	XXXXXXXXX XXXX XXX	DACC02100
C	X X X X X XXXXXXXXX	DACC02110
C	X X X X X X X X	DACC02120
C	XXXXXXXXXXXXXXXXXXXX X X X X X	DACC02130
C	X X X X X X XXXXXXXXXX	DACC02140
C	X X XXXX XXX X	DACC02150
C	X X XXX	DACC02160
C	XXXXXXXXXX	DACC02170
C		DACC02180
C	ARE TYPES OF REGIONS WHICH CONTAIN ONLY ONE COMPONENT BUT WHICH	DACC02190
C	THE PROGRAM CAN INTERPRET AS HAVING TWO OR THREE CLOSED COMPONENTS	DACC02200
C		DACC02210
	CLOSSES = 0	DACC02220
	CMPS = 0	DACC02230
	J1 = END + 1	DACC02240
	END1 = END	DACC02250
	DO 1300 K1 = 1,END1	DACC02260
	HEAD = ELIST (K1,3)	DACC02270
	DO 1300 I = J1,NBLS	DACC02280
	IF (ELIST(K1,3).EQ.ELIST(I,2)) GO TO 2000	DACC02290
1300	CONTINUE	DACC02300
	DO 1350 K1 = 1,END1	DACC02310
	HEAD = ELIST (K1,2)	DACC02320
	DO 1350 I = J1,NBLS	DACC02330
	II = I + 1	DACC02340
	IF (I.EQ.NBLS) II = 1	DACC02350
	IF (ELIST(K1,3).EQ.ELIST(II,3)) GO TO 1990	DACC02360
1350	CONTINUE	DACC02370
	CLOSSES = 1	DACC02380
	CMPS = 1	DACC02390
C		DACC02400
C		DACC02410
1400	END = END + 1	DACC02420
	BEGIN = END	DACC02430
	HEAD = ELIST(BEGIN,2)	DACC02440
	TAIL = ELIST(END,3)	DACC02450
	GO TO 1000	DACC02460



1990	TEMP = ELIST(I,2)	DACC02470
	ELIST(I,2) = ELIST(I,3)	DACC02480
	ELIST(I,3) = TEMP	DACC02490
	TEMPY = ELIST(I,5)	DACC02500
	TEMPX = ELIST(I,4)	DACC02510
	ELIST(I,4) = ELIST(I1,4)	DACC02520
	ELIST(I,5) = ELIST(I1,5)	DACC02530
	ELIST(I1,4) = TEMPX	DACC02540
	ELIST(I1,5) = TEMPY	DACC02550
	RVSL = RVSL + 1	DACC02560
2000	END = END + 1	DACC02570
	IF( END .EQ. I ) GO TO 2050	DACC02580
	DO 2010 K=1,5	DACC02590
2010	HOLDER(K) = ELIST(I,K)	DACC02600
	TEMP = I	DACC02610
2020	TEMP = TEMP - 1	DACC02620
	DO 2025 K=1,5	DACC02630
2025	ELIST(TEMP+1,K) = ELIST(TEMP,K)	DACC02640
	IF( TEMP .GT. END ) GO TO 2020	DACC02650
	DO 2030 K=1,5	DACC02660
2030	ELIST(END,K) = HOLDER(K)	DACC02670
2050	TAIL = ELIST(END,3)	DACC02680
	GO TO 1000	DACC02690
2990	TEMP = ELIST(I,2)	DACC02700
	ELIST(I,2) = ELIST(I,3)	DACC02710
	ELIST(I,3) = TEMP	DACC02720
	TEMPY = ELIST(I,5)	DACC02730
	TEMPX = ELIST(I,4)	DACC02740
	ELIST(I,4) = ELIST(I1,4)	DACC02750
	ELIST(I,5) = ELIST(I1,5)	DACC02760
	ELIST(I1,4) = TEMPX	DACC02770
	ELIST(I1,5) = TEMPY	DACC02780
	RVSL = RVSL + 1	DACC02790
3000	END = END + 1	DACC02800
	DO 3010 K=1,5	DACC02810
3010	HOLDER(K) = ELIST(END,K)	DACC02820
	TEMP = END	DACC02830
3020	TEMP = TEMP - 1	DACC02840
	DO 3030 K=1,5	DACC02850
3030	ELIST(TEMP+1,K) = ELIST(TEMP,K)	DACC02860
	IF( TEMP .GT. BEGIN ) GO TO 3020	DACC02870
	IF( I .EQ. END ) I = BEGIN	DACC02880
	DO 3100 K=1,5	DACC02890
	ELIST(BEGIN,K) = ELIST(I,K)	DACC02900
3100	ELIST(I,K) = HOLDER(K)	DACC02910
	HEAD = ELIST(BEGIN,2)	DACC02920
	GO TO 1000	DACC02930
	END	DACC02940
	SUBROUTINE ADJUST( NELS, CX, CY, NODE )	DACC02950
C		DACC02960
C	ROUTINE TO ADJUST BAD CENTROIDS	DACC02970
C	TO NEAREST NODE	DACC02980
C		DACC02990
	IMPLICIT INTEGER (B-Y)	DACC03000
	DOUBLE PRECISION ZMULT	DACC03010
	COMMON ELIST( 2000, 5 ), ZMULT	DACC03020
C		DACC03030
	ND = 0	DACC03040
	DO 101 I = 1,NELS	DACC03050
	DX = CX-ELIST(I,4)	DACC03060
	DY = CY-ELIST(I,5)	DACC03070

AXY = DX * DX + DY * DY	DACC03080
IF( I .EQ. 1 ) GO TO 100	DACC03090
IF( A .LT. AXY ) GO TO 101	DACC03100
100 A = AXY	DACC03110
ND = I	DACC03120
101 CONTINUE	DACC03130
IF( ND .LE. C ) GO TO 200	DACC03140
CX = ELIST(ND,4)	DACC03150
CY = ELIST(ND,5)	DACC03160
NODE = ELIST(ND,2)	DACC03170
RETURN	DACC03180
C	DACC03190
200 CX = 0	DACC03200
CY = 0	DACC03210
NODE = 0	DACC03220
RETURN	DACC03230
END	CAC 0324C
SUBROUTINE CALC( NSEG, AREA, CX, CY, MINX, MINY )	CAC 03250
C	DACC03260
IMPLICIT INTEGER ( B-Y )	DACC03270
DIMENSION ELIST(2000,5)	DACC03280
COMMON ELIST, ZMULT	DACC03290
DOUBLE PRECISION ZMULT, ARS(X, A, DX,DY, X1,Y1, X2,Y2	DACC03300
DIMENSION ARS(10)	CAC 03310
BEGIN = 1	DACC03320
NAR = 0	DACC03330
A = 0	DACC03340
DX = 0	DACC03350
DY = 0	DACC03360
C	DACC03370
C	DACC03380
C	DACC03390
C	DACC03400
C	DACC03410
MINX = ( ELIST(1,4) / 10000 ) * 10000	CAC 03420
MINY = ( ELIST(1,5) / 10000 ) * 10000	CAC 03421
C	DACC03440
5 NAR = NAR + 1	DACC03450
DO 20 I = BEGIN, NSEG	DACC03460
IF( ELIST(I,1) ) 10,20,20	DACC03470
10 END = I	DACC03480
GO TO 30	DACC03490
20 CONTINUE	DACC03500
END = NSEG	DACC03510
C	DACC03520
C	DACC03530
C	DACC03540
C	DACC03550
C	DACC03560
C	DACC03570
30 X1 = ELIST(END,4) - MINX	DACC03580
Y1 = ELIST(END,5) - MINY	DACC03590
DO 100 K = BEGIN,END	DACC03600
X2 = ELIST( K,4) - MINX	DACC03610
Y2 = ELIST( K,5) - MINY	DACC03620
C	DACC03630
A = A + (X2-X1) * (Y1+Y2)	DACC03640
C	DACC03650
X1 = X2	DACC03660
Y1 = Y2	DACC03670
C	DACC03680

```

100 CONTINUE
C          ACCUMULATE AREAS IN 'ARS'
C
  ARS(NAR) = -A
  A = 0.0000000
C          COMPUTE CENTROID OF LARGEST AREA ONLY
  BEGIN = END + 1
  IF( END .LT. NSEG ) GO TO 5
  BEGIN = 1
  AB = ARS(1)
  IAB = 1
  A = AB
  IF( NAR .EQ. 1 ) GO TO 180
  IF( NAR.GT.10 ) GO TO 500
C
  DO 150 I = 2,NAR
    A = A + ARS(I)
    IF( ARS(I) .LE. AB ) GO TO 150
    AB = ARS(I)
    IAB = I
150  CONTINUE
180  AREA = A * ZMULT
    IF( A .EQ. 0.0 ) GO TO 500
    IF( NAR .EQ. 1 ) GO TO 200
    K = 1
    DO 185 I = 1, NSEG
      IF( ELIST(I,1) .GE. 0 ) GO TO 185
      K = K + 1
      IF( K - IAB )185,183,184
183  BEGIN = I + 1
      GO TO 185
184  END = I
      A = AB
      GO TO 200
185  CONTINUE
      END = NSEG
      A = AB
200  X1 = ELIST(END,4) - MINX
      Y1 = ELIST(END,5) - MINY
      ARSIX = A * 3.0
C
C  NOTE THAT X1 AND Y1 ARE ALREADY SET
C
  DO 300 K = BEGIN,END
    X2 = ELIST( K,4) - MINX
    Y2 = ELIST( K,5) - MINY
C
    DX = DX + ( Y2 - Y1 ) * ( X1*X1 + X1*X2 + X2*X2 ) / ARSIX
    DY = DY + ( X2 - X1 ) * ( Y1*Y1 + Y1*Y2 + Y2*Y2 ) / ARSIX
C
    X1 = X2
    Y1 = Y2
C
300  CONTINUE
C
    CX = DX + MINX
    CY = -DY + MINY
C
C
C
C

```

DACC03690  
DACC03700  
DACC03710  
DACC03720  
DACC03730  
DACC03740  
DACC03750  
DACC03760  
DACC03770  
DACC03780  
DACC03790  
DACC03800  
DACC03810  
DACC03811  
DACC03812  
DACC03820  
DACC03830  
DACC03840  
DACC03850  
DACC03860  
DACC03870  
DACC03880  
DACC03890  
DACC03900  
DACC03910  
DACC03920  
DACC03930  
DACC03940  
DACC03950  
DACC03960  
DACC03970  
DACC03980  
DACC03990  
DACC04000  
DACC04010  
DACC04020  
DACC04030  
DACC04040  
DACC04050  
DACC04060  
DACC04070  
DACC04080  
DACC04090  
DACC04100  
DACC04110  
DACC04120  
DACC04130  
DACC04140  
DACC04150  
DACC04160  
DACC04170  
DACC04180  
DACC04190  
DACC04200  
DACC04210  
DACC04220  
DACC04230  
DACC04240  
DACC04250  
DACC04260  
DACC04270

ORIGINAL PAGE IS  
OF POOR QUALITY

C	RETURN	DACC04280
500	CX = 0	DACC04290
	CY = 0	DACC04300
	RETURN	DACC04310
	END	DACC04320
	SUBROUTINE REFPT(NBLS,KEY)	DAC 04330
	IMPLICIT INTEGER (A-Z)	DAC 04340
	DIMENSION ELIST(2000,5)	DACC04350
	COMMON ELIST	DACC04360
	KEY1=KEY/10000	DACC04370
	KEY2=MOD(KEY,10000)	DACC04380
	WRITE(13,100) KEY1,KEY2	DACC04390
100	FORMAT(1X,2I5)	DACC04400
	XMIN=ELIST(1,4)	DACC04410
	XMAX=ELIST(1,4)	DACC04420
	YMIN=ELIST(1,5)	DACC04430
	YMAX=ELIST(1,5)	DACC04440
	DO 10 J=2,NBLS	DACC04450
	IF(XMIN.GT.ELIST(J,4)) XMIN=ELIST(J,4)	DACC04460
	IF(XMAX.LT.ELIST(J,4)) XMAX=ELIST(J,4)	DACC04470
	IF(YMIN.GT.ELIST(J,5)) YMIN=ELIST(J,5)	DACC04480
	IF(YMAX.LT.ELIST(J,5)) YMAX=ELIST(J,5)	DACC04490
10	CONTINUE	DACC04500
	WRITE(13,200) XMIN,YMIN,XMAX,YMAX,NBLS	DACC04510
200	FORMAT(4I9,15)	DACC04520
	M2=0	DACC04530
	MK=NBLS	DACC04540
20	CONTINUE	DACC04550
	M1=M2+1	DACC04560
	M2=M2+4	DACC04570
	IF(MK.LT.4) M2=NBLS	DACC04580
	WRITE(13,300) (ELIST(M,4),ELIST(M,5),M=M1,M2)	DACC04590
300	FORMAT(4(2I9))	DACC04600
	MK=MK-4	DACC04610
	IF(MK.GE.0) GO TO 20	DACC04620
	RETURN	DACC04630
	END	DACC04640
C	SUBROUTINE INSECT (IXY, IND)	DAC 04650
C	SUBROUTINE TO CHECK TWO LINE SEGMENTS FOR INTERSECTION	DAC 04660
C	THE ENDPPOINTS OF THE LINES ARE TRANSMITTED (INTEGER BINARY)	DACC04670
C	IN THE ARRAY 'IXY' (FIRST 8 ELEMENTS) ( X-Y X-Y X-Y X-Y )	DACC04680
C	THE VARIABLE 'IND' IS RETURNED: 0 IF NO INTERSECTION	DACC04690
C	-1 IF LINES ARE COINCIDENT	DACC04700
C	+1 IF THEY INTERSECT	DACC04710
C	THE COORDINATES OF INTERSECTION	DACC04720
C	ARE RETURNED IN IXY(9) AND IXY(10)	DACC04730
C		DACC04740
C		DACC04750
C		DACC04760
C		DACC04770
C		DACC04780
C	DIMENSION IXY (10),XY(8),S(2),P(2)	DACC04790
C	IND = 0	DACC04800
C	X = 0	DACC04810
C	Y = 0	DACC04820
C		DACC04830
C	RETURN IF NO INTERSECTION POSSIBLE	DACC04840
C		DACC04850
C		DACC04860
C		DACC04870
C		DACC04880

CROSS PRODUCT CALCULATION

ACX = IXY(1) - IXY(5)  
ACY = IXY(2) - IXY(6)

ADX = IXY(1) - IXY(7)	DACC04890
ADY = IXY(2) - IXY(8)	DACC04900
C	DACC04910
BCX = IXY(3) - IXY(5)	DACC04920
BCY = IXY(4) - IXY(6)	DACC04930
BDX = IXY(3) - IXY(7)	DACC04940
BDY = IXY(4) - IXY(8)	DACC04950
A1 = (ACX * ADY - ACY * ADX)	DACC04960
A2 = (BCX * BDY - BCY * BDX)	DACC04970
IF (A1 * A2 .GT. 0.0) RETURN	DACC04980
R = (ACX * BCY - ACY * BCX) * (ADX * BDY - ADY * BDX)	DACC04990
IF (R .GT. 0.0) RETURN	DACC05000
C CHECK COLLINEARITY	DACC05010
IF (A1.EQ.0 .AND. A2.EQ. 0) GO TO 2220	DACC05020
C CALCULATE INTERSECTION	DACC05030
R = 0.0	DACC05040
IF (A1 .NE. 0) R= 1./(1. + ABS(A2/A1))	DACC05050
X = IXY(1) + (IXY(3) - IXY(1)) * R	DACC05060
Y = IXY(2) + (IXY(4) - IXY(2)) * R	DACC05070
C ROUND TO NEAREST INTEGER	DACC05080
IXY(9) = X + SIGN(.5,X)	DACC05090
IXY(10) = Y + SIGN(.5,Y)	DACC05100
IND = 1	DACC05110
RETURN	DACC05120
C COLINEAR CHECK FOR OVERLAP (COINCIDENCE)	DACC05130
2220 IF (ACX * BCX .LT. 0) IND = -1	DACC05140
IF (ADX * BDX .LT. 0) IND = -1	DACC05150
RETURN	DACC05160
END	DACC05170
SUBROUTINE POLYPT (N, NX, NY, IND, MINX, MINY)	DACC05180
C	DACC05190
C ROUTINE TO DETERMINE WHETHER A POINT IS WITHIN A POLYGON	DACC05200
C	DACC05210
C THE POLYGON BOUNDARY SEGMENTS ARE IN ELIST(1,4), ELIST(1,5)	DACC05220
C THE NUMBER OF SEGMENTS IS 'N'	DACC05230
C	DACC05240
C 'IND' IS RETURNED: +1 IF PT. IS INSIDE	DACC05250
C 0 IF PT. IS ON THE BOUNDARY	DACC05260
C -1 IF PT. IS OUTSIDE	DACC05270
C	DACC05280
DIMENSION LXY(10),ELIST (2000,5)	DACC05290
COMMON ELIST, ZMLLT	DACC05300
INTEGER ELIST	DACC05310
DOUBLE PRECISION ZMULT	DACC05320
IX = NX - MINX + 1	DACC05330
IY = NY - MINY + 1	DACC05340
IND = 0	DACC05350
INDX = 0	DACC05360
XX=100000.	DACC05370
KOUNT = KOUNT + 1	DACC05380
LXY(1) = 0	DACC05390
LXY(2) = IY	DACC05400
LXY(3) = IX	DACC05410
LXY(4) = IY	DACC05420
LXY(9) = 0	DACC05430
LXY(10)= 0	DACC05440
IT = 1	DACC05450
GO TO 2000	DACC05460
C	DACC05470
C	DACC05480
C	DACC05490

C	DRAW A LINE FROM THE POINT TO THE Y-AXIS AND COUNT THE NUM	DACC05500
C	OF INTERSECTIONS WITH BOUNDARY SEGMENTS. ODD IS INSIDE, EVEN	DACC05510
C	OUTSIDE	DACC05520
C	IF AN INTERSECTION OCCURS AT A NODE	DACC05530
C	CHANGE THE Y COORDINATE AT THE AXIS AND START OVER	DACC05540
C		DACC05550
	1000 IT = IT + 1	DACC05560
	LXY (2) = LXY (2) + IY/10 + 1	DACC05570
	IF (IT.GT.5) RETURN	DACC05580
	2000 INDX = 0	DACC05590
	DO 3000 I = 1, N	DACC05600
	I1 = I + 1	DACC05610
	IF (I.EQ.N) I1 = 1	DACC05620
	LXY (5) = ELIST(I,4) - MINX + 1	DACC05630
	LXY (6) = ELIST(I,5) - MINY + 1	DACC05640
	LXY (7) = ELIST(I1,4) - MINX + 1	DACC05650
	LXY (8) = ELIST(I1,5) - MINY + 1	DACC05660
	CHECK FOR PT AT A NODE	DACC05670
C		DACC05680
	DO 2110 L=5,7,2	DACC05690
	IF (LXY(L).EQ.IX.AND.LXY(L+1).EQ.IY) RETURN	DACC05700
	2110 CONTINUE	DACC05710
	CALL INSECT( LXY, INT )	DACC05720
	IF (INT) 1000,3000,2400	DACC05730
C		DACC05740
	CHECK FOR PT. ON BOUNDARY	DACC05750
C		DACC05760
	2400 IF (LXY(9).EQ.IX.AND.LXY(10).EQ.IY) RETURN	DACC05770
C		DACC05780
	CHECK FOR INTERSECTION WITH A CORNER	DACC05790
C		DACC05800
	DO 2500 J=6,8,2	DACC05810
	IF( LXY(9) .NE. LXY(J-1) ) GO TO 2500	DACC05820
	IF (LXY(10).EQ.LXY(J)) GO TO 1000	DACC05830
	2500 CONTINUE	DACC05840
	INDX = INDX + 1	DACC05850
	3000 CONTINUE	DACC05860
	IND = 1	DACC05870
	IF((INDX/2) * 2 .EQ. INDX ) IND = -1	DACC05880
	RETURN	DACC05890
	END	DACC05900
	SUBROUTINE ADJNCY (NBLS, KEY,LIST )	DACC05910
C		DACC05920
C	ADJACENCY LIST	DACC05930
C		DACC05940
	DIMENSION LIST(2000)	DACC05950
	DATA NOUT3 / 12 /	DACC05960
	NEND = NBLS - 1	DACC05970
	DO 1200 I = 1, NEND	DACC05980
	IF( LIST(I) .LT. 0 ) GO TO 1200	DACC05990
	NEXT= I + 1	DACC06000
	N = LIST(I)	DACC06010
C		DACC06020
	UNDUPLICATE LIST	DACC06030
	DO 1100 J = NEXT, NBLS	DACC06040
	IF( LIST(J) .EQ. N ) LIST(J) = -1	DACC06050
	1100 CONTINUE	DACC06060
	1200 CONTINUE	DACC06070
C		DACC06080
	PRINT AND COPY LIST	DACC06090
	WRITE(6, 1300 ) KEY	DACC06100
	1300 FORMAT( '0 LIST OF ADJACENT AREAS FOR', I11/ )	
	DO 1500 I = 1, NBLS	

IF( LIST(I) .LE. 0 ) GO TO 1500  
WRITE(6,1400) LIST(I)  
1400 FORMAT( 50X, I11 )  
WRITE(NDUT3,1450 ) KEY, LIST(I)  
1450 FORMAT( 2I15 )  
C  
1500 CONTINUE  
RETURN  
END

DACC06110  
DACC06120  
DACC06130  
DACC06140  
DACC06150  
DACC06160  
DACC06170  
DACC06180  
DAC 06190

ORIGINAL PAGE IS  
OF POOR QUALITY

APPENDIX VII  
PIOS PROGRAM LISTING



C	CORRECTED VERSION OF PIOS/75 WITH FIVE (5) FILES	00010000
C	DRIVER/LA	00020000
C	LATEST UPDATE JUNE 23 '75	00030000
	INTEGER ENDOF(20)	00040000
	DATA ENDOF/'9999',*9 *.12* '/'	00050000
	DIMENSION P(3000),Q(3000)	00060000
	DIMENSION X(3000),Y(3000),XLE(1300),YLE(1300),XHE(1300)	00070000
	DIMENSION YHE(1300),IRE(1300)	00080000
	DIMENSION INTSP(20,2)	00090000
	DIMENSION NSTPL(16),LSTP(16,2,50),XSTR(3000),YSTR(3000)	00100000
	DIMENSION BL(16),BH(16)	00110000
	REAL*8 A,B,C,D,E,F,DET,P0,T0	00120000
	COMMON /ALPHA/IV1,IV2,IV3,IV4,IV5	00130000
	REAL LUAREA	00140000
	INTEGER MAPND,MAPOL1(2),LUCODE(3)	00150000
	INTEGER SPMAP,SPPOLY(2),SPCODE(3)	00160000
	INTEGER FILL/' '/'	00170000
C	LOGICAL DONUT/.FALSE./,OUT/.FALSE./	00180000
	LOGICAL IFLAG	00190000
	LOGICAL ALLIN,ALLOUT,FLAG,PQ(3000),DIAG,PRTOUT/.FALSE./	00200000
	ITERRY=0	00210000
	DEFINE FILE 11(3000,80,E,IV1)	00220000
	DEFINE FILE 12(3000,80,E,IV2)	00230000
	DEFINE FILE 13(3000,39,E,IV3)	00240000
	DEFINE FILE 14(3000,80,E,IV4)	002500
	DEFINE FILE 15(3000,31,E,IV5)	00260000
C	*****	002700
C		00280000
C		00290000
C		00300000
C	* * * * *	00310000
C	MAIN PROGRAM FOR POLYGON OVERLAY	00320000
C	* * * * *	00330000
C		00340000
C	READ TEST POLYGON	00350000
C		00360000
C	MAX - THE NUMBER OF POINTS IN THE POLYGON	00370000
C	YMAX - THE MAXIMUM Y VALUE	00380000
C	YMIN - THE MINIMUM Y VALUE	00390000
C	X - X-COORDINATES	00400000
C	Y - Y-COORDINATES	00410000
C		00420000
C	DEFINE I/O UNITS	00430000
C		00440000
C		00450000
C	BE SURE TO EXPAND ALL ARRAYS AND DEFINE FILES TO	00460000
C	ACCOMMODATE YOUR NEEDS	00470000
C		00480000
C	ONLY ONE CARD IMAGE IS READ IN ON UNIT 5; PRTOUT	00490000
C	.TRUE. IN COLS COLS 1-6 WILL RESULT IN FILE 14 BEING PRINTED OUT	00500000
C	1 IN COLS COL 1 WILL RESULT IN FILE 14 BEING PRINTED OUT	00510000
C	0 IN COL COL 1 WILL RESULT IN FILE 14 NOT BEING PRINTED OUT	00520000
C		00530000
C	READ (5,20)PRTOUT	00540000
C	20 FORMAT (11)	00550000
	IV1=1	00560000
	IV2=1	00570000
	IV3=1	00580000
	IV4=1	00590000
	IV5=1	00600000
	FIND (13'1)	00610010

ORIGINAL PAGE IS  
OF POOR QUALITY

	FIND (14'1)	00620020
	FIND (15'1)	00630030
	WRITE (6,2700)	00640000
2700	FORMAT (' CLUGE CALLED BY DRIVER/LA EMG     ')	00650000
	CALL CLUGE	00660000
	WRITE (6,2710)	00670000
2710	FORMAT (' RETURNED FROM CLUGE EMG    ')	00680000
	FIND (11'1)	00690000
	FIND (12'1)	00700000
	IV1=1	00710000
	IV2=1	00720000
C	CLUGE USED ONLY FOR TEST CASE	00730000
C	BRINGS IN TEST DATA	00740000
	IUNIT=5	00750000
	JUNIT=6	00760000
	INTZ=11	00770000
	INSP=12	00780000
	IASC=6	00790000
	IZZ=13	00800000
C		00810000
C	SET SWITCH FOR DIAGNOSTIC PRINT OUT	00820000
C		00830000
	ICNT=1	00840000
	DO 50 I=1,1300	00850000
	30 READ (12'IV2,40,ERR=70) MAPOL1,XLE(I),YLE(I),XHE(I),	008600
	1YHE(I),N,(X(K),Y(K),K=1,N)	00870000
	40 FORMAT (2I5/4F9.0,15/(8F9.0))	008800
	IF (N.LE.3000)GO TO 45	00890000
	42 FORMAT (' MAPNO, N= ',2(I5,2X))	00900050
	WRITE(6,44) MAPOL1,N	009100
	44 FORMAT (' ----- ERROR---ERROR---TOO MANY POINTS',/, ' ',15,2X,	00920000
	115, ' N= ',16, ' PIOS/II TERMINATED')	009300
	STOP	00940000
	45 NMPOLY=I	00950000
	WRITE(6,42) MAPOL1 (1),N	009600
	MAPNO = MAPOL1 (1)	009601
C	CALCULATE RECORD COUNT	00970000
	IRE(I)=ICNT	00980000
	ICNT=ICNT+2+(N+3)/4	00990000
	IF (MAPNO.EQ.99999) GO TO 70	01000000
	50 CONTINUE	01010000
	WRITE (6,60)	01020000
	60 FORMAT (1X, ' MORE THAN 1300 POLYGONS-- PROCESS FIRST 1300.')	01030000
	70 CONTINUE	01040000
	DIAG=.FALSE.	01050000
	80 CONTINUE	01060000
C		01070000
C	SEARCH FOR SPECIFIED TRAFFIC ZONE	01080000
C		01090000
C		01100000
C	READ MAJOR POLYGON	01110000
C		01120000
	CALL POLYP? (8930,MAPNO,MAPOL1,LUCODE,LUAREA,XMIN,YMIN,XMAX,YMAX,M	01130000
	1AX,X,Y,INTZ,FAC,DONUT,OUT,IV1)	01140000
	WRITE(15'IV5,86) MAPNO,MAPOL1(1),LUCODE(3),MAPOL1(2),LUAREA	011500
	WRITE(7,86) MAPNO,MAPOL1(1),LUCODE(3),MAPOL1(2),LUAREA	011501
	86 FORMAT (2I5,I4,I2,F15.2)	011502
	FIND (15'IV5)	01170000
C		01180000
C	SET THE SIGNIFIGANCE VALUE FOR THE	01190000
C	APPROXIMATION FOR THE INTERSECTIONS	01200000

C		01210000
	SIG=.01	01220000
C		01230000
C	INITIALIZE THE UPPER AND LOWER	01240000
C	BOUNDS FOR XX AND YY.	01250000
	YY=0.0	01260000
	XX=1.0E6	01270000
	M=MAX-1	01280000
C		01290000
C	FIND THE LOWER LEFT VERTEX OF	01300000
	THE MAJOR POLYGON.	01310000
	DO 110 J=1,M	01320000
	IF (Y(J)-YY) 110,100,90	01330000
90	YY=Y(J)	01340000
	XX=X(J)	01350000
	IJ=J	01360000
	GO TO 110	01370000
100	IF (X(J)-XX) 90,110,110	01380000
110	CONTINUE	01390000
C		01400000
C	OUTPUT DIAGNOSTICS IF REQUESTED	01410000
C		01420000
C	COMPUTE AND PRINT LEVELS	01430000
C		01440000
C	SET VALUE OF THE STRIPPING	01450000
C	EXPONENT	01460000
	NP=2	014700
	ISTP=2**NP	01480000
	STEP=(YMAX-YMIN)/ISTP	01490000
	BLOW=YMIN-STEP	01500000
	DO 120 J=1,ISTP	01510000
	BLOW=BLOW+STEP	01520000
	BHIH=BLOW+STEP	01530000
	BL(J)=BLOW	01540000
	BH(J)=BHIH	01550000
120	CONTINUE	01560000
	BH(ISTP)=YMAX	01570000
C		01580000
C	INITIALIZE LEVEL AND STRING LISTS	01590000
C		01600000
	DO 130 IC=1,16	01610000
	NSTPL(IC)=0	01620000
	DO 130 IA=1,20	01630000
	DO 130 IB=1,2	01640000
130	LSTP(IC,IB,IA)=0	01650000
	ISTRG=0	01660000
C		01670000
C	START STRING ASSIGNMENTS	01680000
C		01690000
	XLAST=X(IJ)	01700000
	YLAST=Y(IJ)	01710000
	NDX=IJ	01720000
	LLEV=LEVEL(YLAST,YMIN,STEP,ISTP,YMAX)	01730000
	ISTRG=ISTRG+1	01740000
	XSTR(ISTRG)=XLAST	01750000
	YSTR(ISTRG)=YLAST	01760000
	NSTPL(LLEV)=NSTPL(LLEV)+1	01770000
	ILS=NSTPL(LLEV)	01780000
	NF=ISTRG	01790000
C		01800000
C	LOOP THRU POINTS	01810000

C	DO 310 J=2,MAX	01820000
	NDX=MOD(NDX,M)+1	01830000
	XCUR=X(NDX)	01840000
	YCUR=Y(NDX)	01850000
	LEVC=LEVEL(YCUR,YMIN,STEP,ISTP,YMAX)	01860000
	IF (LLEV.NE.LEVC) GO TO 160	01870000
C		01880000
C	SAME LEVEL - SAME STRING	01890000
C		01900000
	IF (YCUR.NE.YSTR(ISTRG)) GO TO 140	01910000
	IF (XCUR.NE.XSTR(ISTRG)) GO TO 140	01920000
	GO TO 150	01930000
140	ISTRG=ISTRG+1	01940000
	XSTR(ISTRG)=XCUR	01950000
	YSTR(ISTRG)=YCUR	01960000
150	CONTINUE	01970000
	GO TO 300	01980000
		01990000
C		02000000
C	DIFFERNT LEVEL	02010000
C		02020000
160	CONTINUE	02030000
C		02040000
C	DEFINE DIRECTION	02050000
C		02060000
	IDIR=LEVC-LLEV	02070000
	IF (IDIR.LT.0) GO TO 220	02080000
		02090000
C	POSITIVE DIRECTION	02100000
C	CLOSE LAST LEVEL	02110000
C		02120000
170	YTEMP=BH(LLEV)	02130000
	XTEMP=(XCUR-XLAST)*(YTEMP-YLAST)/(YCUR-YLAST)+XLAST	02140000
	IF (YTEMP.NE.YSTR(ISTRG)) GO TO 180	02150000
	IF (XTEMP.NE.XSTR(ISTRG)) GO TO 180	02160000
	GO TO 190	02170000
180	ISTRG=ISTRG+1	02180000
	XSTR(ISTRG)=XTEMP	02190000
	YSTR(ISTRG)=YTEMP	02200000
190	CONTINUE	02210000
	IF (NF.NE.ISTRG) GO TO 200	02220000
	NSTPL(LLEV)=NSTPL(LLEV)-1	02230000
	GO TO 210	02240000
200	LSTP(LLEV,1,ILS)=NF	02250000
	LSTP(LLEV,2,ILS)=ISTRG	02260000
210	CONTINUE	02270000
C	START NEW LEVEL	02280000
	LLEV=LLEV+1	02290000
	NSTPL(LLEV)=NSTPL(LLEV)+1	02300000
	ILS=NSTPL(LLEV)	02310000
	NF=ISTRG	02320000
	IF (LLEV.LT.LEVC) GO TO 170	02330000
	GO TO 270	02340000
		02350000
C		02360000
C	NEGATIVE DIRECTION	02370000
C	CLOSE LAST LEVEL	02380000
C		02390000
220	YTEMP=BL(LLEV)	02400000
	XTEMP=(XCUR-XLAST)*(YTEMP-YLAST)/(YCUR-YLAST)+XLAST	02410000
	IF (YTEMP.NE.YSTR(ISTRG)) GO TO 230	02420000
	IF (XTEMP.NE.XSTR(ISTRG)) GO TO 230	

GO TO 240	02430000
230 ISTRG=ISTRG+1	02440000
XSTR(ISTRG)=XTEMP	02450000
YSTR(ISTRG)=YTEMP	02460000
240 CONTINUE	02470000
IF (NF.NE.ISTRG) GO TO 250	02480000
NSTPL(LLEV)=NSTPL(LLEV)-1	02490000
GO TO 260	02500000
250 LSTP(LLEV,1,ILS)=NF	02510000
LSTP(LLEV,2,ILS)=ISTRG	02520000
260 CONTINUE	02530000
C	02540000
C START NEW LEVEL	02550000
C	02560000
LLEV=LLEV-1	02570000
NSTPL(LLEV)=NSTPL(LLEV)+1	02580000
ILS=NSTPL(LLEV)	02590000
NF=ISTRG	02600000
IF (LLEV.GT.LEVC) GO TO 220	02610000
GO TO 270	02620000
270 IF (YCUR.NE.YSTR(ISTRG)) GO TO 280	02630000
IF (XCUR.NE.XSTR(ISTRG)) GO TO 280	02640000
GO TO 290	02650000
280 ISTRG=ISTRG+1	02660000
XSTR(ISTRG)=XCUR	02670000
YSTR(ISTRG)=YCUR	02680000
290 CONTINUE	02690000
300 XLAST=XCUR	02700000
YLAST=YCUR	02710000
310 CONTINUE	02720000
C	02730000
C CLOSE LAST STRING	02740000
C	02750000
IF (NF.NE.ISTRG) GO TO 320	02760000
NSTPL(LLEV)=NSTPL(LLEV)-1	02770000
GO TO 330	02780000
320 LSTP(LLEV,1,ILS)=NF	02790000
LSTP(LLEV,2,ILS)=ISTRG	02800000
330 CONTINUE	02810000
C	02820000
C LOOP ON MINOR POLYGONS	02830000
C	02840000
C	02850000
C *****	02860000
C READ MINOR POLYGON	02870000
C *****	02880000
C	02890000
C	02900000
IOVER=0	02910000
DO 910 IW=1,NMPOLY	02920000
C CHECK AND SEE IF EVEN ONE CORNER IS IN THIS MAJOR POLYGON	02930000
IF (YHE(IW).LT.YMIN) GO TO 910	02940000
IF (YLE(IW).GT.YMAX) GO TO 910	02950000
IF (XHE(IW).LT.XMIN) GO TO 910	02960000
IF (XLE(IW).GT.XMAX) GO TO 910	02970000
C IF GET THIS FAR, THEN HAVE AN OVERLAP.	02980000
NN=IRE(IW)	02990000
FIND (12*NN)	03000000
IV2=NN	03010050
CALL POLYRD (910,SPMAP,SPPOLY,SPCODE,SPAREA,PXMIN,PYMIN,PXMAX,PYM03020000	03030000
IAX,NPTS,P,Q,INSP,FAC,DONUT,OUT,IV2)	

C		03040000
C	IS MINOR POLYGON OUTSIDE OF TRAFFIC ZONE	03050000
340	ALLIN=.TRUE.	03060000
	ALLOUT=.TRUE.	03070000
C		03080000
	SPCODE(2)=LUCODE(2)	03090000
	DO 490 I=1,NPTS	03100000
	PP=P(I)	03110000
	QQ=Q(I)	03120000
	IF (QQ.LT.YMIN) GO TO 470	03130000
	IF (QQ.GT.YMAX) GO TO 470	03140000
	IF (PP.LT.XMIN) GO TO 470	03150000
	IF (PP.GT.XMAX) GO TO 470	03160000
	J=LEVEL(QQ,YMIN,STEP,ISTP,YMAX)	03170000
	JK=NSTPL(J)	03180000
	FLAG=.FALSE.	03190000
	IF (JK.EQ.0) GO TO 460	03200000
	DO 450 JJ=1,JK	03210000
	YREM=-1.0	03220000
	JL=LSTP(J,1,JJ)	03230000
	JM=LSTP(J,2,JJ)-1	03240000
	DO 450 JN=JL,JM	03250000
	YP=YSTR(JN)	03260000
	YYP=YSTR(JN+1)	03270000
	XP=XSTR(JN)	03280000
	XXP=XSTR(JN+1)	03290000
	IF (QQ.GT.YP.AND.QQ.GT.YYP) GO TO 440	03300000
	IF (QQ.LT.YP.AND.QQ.LT.YYP) GO TO 440	03310000
	IF (PP.GT.XP.AND.PP.GT.XXP) GO TO 440	03320000
	IF (QQ.NE.YP) GO TO 370	03330000
	IF (PP.EQ.XP) GO TO 430	03340000
	IF (QQ.EQ.YREM) GO TO 400	03350000
	YREM=YP	03360000
	LAST=JN	03370000
350	LAST=LAST-1	03380000
	IF (LAST.EQ.0) GO TO 450	03390000
	IF (QQ.EQ.YSTR(LAST)) GO TO 350	03400000
	YLAST=YSTR(LAST)	03410000
	NEXT=JN	03420000
360	NEXT=NEXT+1	03430000
	IF (NEXT.GT.ISTRG) GO TO 450	03440000
	IF (QQ.EQ.YSTR(NEXT)) GO TO 360	03450000
	YNEXT=YSTR(NEXT)	03460000
	IF (QQ.GT.YLAST.AND.QQ.GT.YNEXT) GO TO 450	03470000
	IF (QQ.LT.YLAST.AND.QQ.LT.YNEXT) GO TO 450	03480000
	IF (PP.LT.XP) GO TO 420	03490000
	GO TO 450	03500000
370	IF (QQ.NE.YYP) GO TO 410	03510000
	IF (PP.EQ.XXP) GO TO 430	03520000
	IF (QQ.EQ.YREM) GO TO 400	03530000
	YREM=YYP	03540000
	LAST=JN+1	03550000
380	LAST=LAST-1	03560000
	IF (QQ.EQ.YSTR(LAST)) GO TO 380	03570000
	YLAST=YSTR(LAST)	03580000
	NEXT=JN+1	03590000
390	NEXT=NEXT+1	03600000
	IF (QQ.EQ.YSTR(NEXT)) GO TO 390	03610000
	YNEXT=YSTR(NEXT)	03620000
	IF (QQ.GT.YLAST.AND.QQ.GT.YNEXT) GO TO 450	03630000
	IF (QQ.LT.YLAST.AND.QQ.LT.YNEXT) GO TO 450	03640000

ORIGINAL PAGE IS  
OF POOR QUALITY

IF (PP.LE.XXP) GO TO 420	03650000
GO TO 450	03660000
400 IF (YP.NE.YYP) YREM=-1.0	03670000
GO TO 450	03680000
410 YREM=-1.0	03690000
IF (PP.LE.XP.AND.PP.LE.XXP) GO TO 420	03700000
IF (XP.EQ.XXP) GO TO 430	03710000
IF (YP.EQ.YYP) GO TO 430	03720000
XTEMP=(XP-XXP)*(QQ-YYP)/(YP-YYP)+XXP	03730000
IF (PP.GT.XTEMP) GO TO 450	03740000
IF (PP.EQ.XTEMP) GO TO 430	03750000
420 FLAG=.NOT.FLAG	03760000
GO TO 450	03770000
430 FLAG=.TRUE.	03780000
GO TO 460	03790000
440 YREM=-1.0	03800000
450 CONTINUE	03810000
460 CONTINUE	03820000
IF (FLAG) GO TO 480	03830000
470 PQ(I)=.FALSE.	03840000
ALLIN=.FALSE.	03850000
GO TO 490	03860000
480 PQ(I)=.TRUE.	03870000
ALLOUT=.FALSE.	03880000
490 CONTINUE	03890000
IF (ALLOUT) WRITE (6,500) SPPOLY,SPCODE	03900000
500 FORMAT (1X,'ALLOUT',15,1X,12,1X,A3,1X,15,1X,14)	03910000
IF (ALLIN) GO TO 880	03920000
IF (ALLOUT) GO TO 900	03930000
C	03940000
C *	03950000
C BUILD THE INTERSECTING POLYGONS	03960000
C *	03970000
C	03980000
C AN ARRAY IS CONSTRUCTED THAT CONTAINS ALL THE POINTS OF THE	03990000
C INTERSECTING POLYGON(S)	04000000
C	04010000
C THE MINOR POLYGON IS SEARCHED IN A SERIAL MANNER. AS EACH PAIR	04020000
C OF POINTS ARE ENCOUNTERED THE FOLLOWING CASES AND ACTIONS	04030000
C OCCUR	04040000
C	04050000
C I-1 I	04060000
C	04070000
C OUT OUT POINT I IS NOT ADDED TO THE ARRAY	04080000
C	04090000
C OUT IN THE MAJOR POLYGON IS SEARCHED TO FIND THE LINE	04100000
C SEGMENT THAT INTERSECTS WITH THE MINOR POLYGON.	04110000
C THIS IS ACCOMPLISHED BY FIRST SEARCHING FOR LINE	04120000
C SEGMENTS THAT INTERSECT WITH THE RECTANGULAR	04130000
C REGION DEFINED BY THE POINTS I-1 AND I. EACH	04140000
C LINE SEGMENT THAT DOES SO IS TESTED FOR INTERSECTION	04150000
C WITH THE LINE FORMED BY I-1 , I. AT LEAST ONE	04160000
C SUCH INTERSECTION MUST OCCUR (OR THE POINT I	04170000
C WOULD BE OUT). THE POINT OF INTERSECTION IS ADDED TO	04180000
C THE ARRAY. STEPPING IS CONTINUED ON THE	04190000
C MINOR POLYGON.	04200000
C	04210000
C IN IN THE POINT I IS ADDED TO THE ARRAY	04220000
C	04230000
C IN OUT THE MAJOR POLYGON IS SEARCHED FOR THE INTERSECTION.	04240000
C THE INTERSECTION POINT IS ADDED TO THE ARRAY.	04250000

C	NOW STEPPING IS PERFORMED ON THE MAJOR POLYGON	04260000
C	IN THE DIRECTION SUCH THAT THE POINTS ARE IN THE	04270000
C	MINOR POLYGON. THESE POINTS ARE ADDED TO THE ARRAY.	04280000
C	WHEN THE STEPPING CAUSES A POINT TO BE OUT OF	04290000
C	THE MINOR POLYGON THE INTERSECTION POINT IS FOUND.	04300000
C	BY SEARCHING THE MINOR POLYGON, AND IS ADDED TO	04310000
C	THE ARRAY. STEPPING THEN RESUMES ON THE MINOR	04320000
C	POLYGON WHERE IT WAS LAST INTERRUPTED.	04330000
C		04340000
	WRITE (6,7001)MAPNO,MAPOL1(1),SPMAP,SPCODE(1),SPCODE(3)	04350000
7001	FORMAT (' START BUILD- MAJOR ',I5,2X,I5,2X,' MINOR ',I5,2X,A3,2X,I4)	04360000
	NINTS=0	04370000
	J=0	04380000
	I=1	04390000
	IF (PQ(I)) GO TO 560	04400000
C	POINT I IS OUT	04410000
510	I=I+1	04420000
	IF (I.GT.NPTS) GO TO 790	04430000
	IF (.NOT.PQ(I)) GO TO 510	04440000
		04450000
C	POINT I-1 WAS OUT. POINT I IS IN	04460000
C	FIND INTERSECTION BY SEARCHING MAJOR POLYGON	04470000
C		04480000
	K=0	04490000
520	K=K+1	04500000
	IF (K.GT.(ISTRG-1)) GO TO 530	04510000
C		04520000
C	FIND SPAN	04530000
C		04540000
	Y1=YSTR(K)	04550000
	Y2=YSTR(K+1)	04560000
	V1=Q(I-1)	04570000
	V2=Q(I)	04580000
	IF (Y1.LT.V1.AND.Y2.LT.V1.AND.Y1.LT.V2.AND.Y2.LT.V2) GO TO 520	04590000
	IF (Y1.GT.V1.AND.Y2.GT.V1.AND.Y1.GT.V2.AND.Y2.GT.V2) GO TO 520	04600000
	X1=XSTR(K)	04610000
	X2=XSTR(K+1)	04620000
	U1=P(I-1)	04630000
	U2=P(I)	04640000
	IF (X1.LT.U1.AND.X2.LT.U1.AND.X1.LT.U2.AND.X2.LT.U2) GO TO 520	04650000
	IF (X1.GT.U1.AND.X2.GT.U1.AND.X1.GT.U2.AND.X2.GT.U2) GO TO 520	04660000
		04670000
C	TEST FOR INTERSECTION	04680000
C		04690000
C		04700000
	A=X2-X1	04710000
	B=U1-U2	04720000
	C=Y2-Y1	04730000
	D=V1-V2	04740000
	DET=A*D-C*B	04750000
	IF (DET.EQ.0) GO TO 520	04760000
	E=U1-X1	04770000
	F=V1-Y1	04780000
	T0=(D*E-B*F)/DET	04790000
	P0=(A*F-C*E)/DET	04800000
	IF (T0.LT.0.0.OR.P0.LT.0.0) GO TO 520	04810000
	IF (T0.GT.1.0.OR.P0.GT.1.0) GO TO 520	04820000
	XINT=T0*X2+(1.0-T0)*X1	04830000
	YINT=T0*Y2+(1.0-T0)*Y1	04840000
	J=J+1	04850000
	X(J)=XINT	04860000
	Y(J)=YINT	



NINTS=NINTS+1	04870000
INTSP(NINTS,1)=J	04880000
GO TO 560	04890000
530 CONTINUE	04900000
IERR=1	04910000
WRITE (6,540) IERR	04920000
540 FORMAT (' COULD NOT FIND INTERSECTION AT CHECK POINT ',I5)	04930000
WRITE (6,550) SPMAP,SPPOLY	04940000
550 FORMAT (IX,I5,IX,A3,IX,I2)	04950000
WRITE (6,620) P(I),Q(I),P(I-1),Q(I-1)	04960000
GO TO 910	04970000
C POINT I IS IN	04980000
560 J=J+1	04990000
X(J)=P(I)	05000000
Y(J)=Q(I)	05010000
IF (I.GT.1) GO TO 570	05020000
NINTS=NINTS+1	05030000
INTSP(NINTS,1)=J	05040000
570 CONTINUE	05050000
I=I+1	05060000
IF (I.GT.NPTS) GO TO 580	05070000
IF (PQ(I)) GO TO 560	05080000
GO TO 590	05090000
580 INTSP(NINTS,2)=J	05100000
GO TO 790	05110000
590 CONTINUE	05120000
C	05130000
C POINT I IS OUT, POINT I-1 WAS IN	05140000
C FIND INTERSECTION BY S MAJOR POLYGON	05150000
C	05160000
K=0	05170000
600 K=K+1	05180000
IF (K.GT.(ISTRG-1)) GO TO 610	05190000
C	05200000
C FIND SPAN	05210000
C	05220000
Y1=YSTR(K)	05230000
Y2=YSTR(K+1)	05240000
V1=Q(I-1)	05250000
V2=Q(I)	05260000
IF (Y1.LT.V1.AND.Y2.LT.V1.AND.Y1.LT.V2.AND.Y2.LT.V2) GO TO 600	05270000
IF (Y1.GT.V1.AND.Y2.GT.V1.AND.Y1.GT.V2.AND.Y2.GT.V2) GO TO 600	05280000
X1=XSTR(K)	05290000
X2=XSTR(K+1)	05300000
U1=P(I-1)	05310000
U2=P(I)	05320000
IF (X1.LT.U1.AND.X2.LT.U1.AND.X1.LT.U2.AND.X2.LT.U2) GO TO 600	05330000
IF (X1.GT.U1.AND.X2.GT.U1.AND.X1.GT.U2.AND.X2.GT.U2) GO TO 600	05340000
C	05350000
C TEST FOR INTERSECTION	05360000
C	05370000
A=X2-X1	05380000
B=U1-U2	05390000
C=Y2-Y1	05400000
D=V1-V2	05410000
DET=A*D-C*B	05420000
IF (DET.EQ.0) GO TO 600	05430000
E=U1-X1	05440000
F=V1-Y1	05450000
T0=(D*E-B*F)/DET	05460000
P0=(A*F-C*E)/DET	05470000

IF (T0.LT.0.0.OR.P0.LT.0.0) GO TO 600	05480000
IF (T0.GT.1.0.OR.P0.GT.1.0) GO TO 600	05490000
XINT=T0*X2+(1.0-T0)*X1	05500000
YINT=T0*Y2+(1.0-T0)*Y1	05510000
J=J+1	05520000
X(J)=XINT	05530000
Y(J)=YINT	05540000
GO TO 630	05550000
610 IERR=2	05560000
WRITE (6,540) IERR	05570000
WRITE (6,550) SPMAP,SPPOLY	05580000
WRITE (6,620) P(I),Q(I),P(I-1),Q(I-1)	05590000
620 FORMAT (4F10.0)	05600000
GO TO 910	05610000
630 CONTINUE	05620000
C	05630000
C	05640000
C	05650000
700 IDIR=1	05660000
K=K+1	05670000
XE=XSTR(K)	05680000
YE=YSTR(K)	05690000
710 XB=XINT	05700000
YB=YINT	05710000
720 CALL STPSUB (XB,YB,XE,YE,SPX,SPY,NOSTP)	05720000
DO 740 L=1,NOSTP	05730000
XB=XB+SPX	05740000
YB=YB+SPY	05750000
CALL PIP (XB,YB,IND,P,Q,NPTS)	05760000
730 IF (IND.EQ.0) GO TO 750	05770000
740 CONTINUE	05780000
C	05790000
IF IT GETS HERE THE END POINT IS IN	05800000
J=J+1	05810000
X(J)=XE	05820000
Y(J)=YE	05830000
C	05840000
TAKE NEXT POINT	05850000
K=K+IDIR	05860000
IF (K.EQ.0) K=ISTRG-1	05870000
IF (K.GT.(ISTRG-1)) K=1	05880000
XB=XE	05890000
YB=YE	05900000
XE=XSTR(K)	05910000
YE=YSTR(K)	05920000
GO TO 720	05930000
750 CONTINUE	05940000
C	05950000
POINT (XB-SPX,YB-SPY) WAS IN AND (XB,YB) IS OUT	05960000
760 L=0	05970000
770 L=L+1	05980000
IF (L.EQ.NPTS) GO TO 780	05990000
Y1=Q(L)	06000000
Y2=Q(L+1)	06010000
V1=YB	06020000
V2=YB-SPY	06030000
IF (Y1.LT.V1.AND.Y2.LT.V1.AND.Y1.LT.V2.AND.Y2.LT.V2) GO TO 770	06040000
IF (Y1.GT.V1.AND.Y2.GT.V1.AND.Y1.GT.V2.AND.Y2.GT.V2) GO TO 770	06050000
X1=P(L)	06060000
X2=P(L+1)	06070000
U1=XB	06080000
U2=XB-SPX	
IF (X1.LT.U1.AND.X2.LT.U1.AND.X1.LT.U2.AND.X2.LT.U2) GO TO 770	
IF (X1.GT.U1.AND.X2.GT.U1.AND.X1.GT.U2.AND.X2.GT.U2) GO TO 770	

C		06090000
C	TEST FOR INTERSECTION	06100000
C		06110000
	A=X2-X1	06120000
	B=U1-U2	06130000
	C=Y2-Y1	06140000
	D=V1-V2	06150000
	DET=A*D-C*B	06160000
	IF (DET.EQ.0.0) GO TO 770	06170000
	E=U1-X1	06180000
	F=V1-Y1	06190000
	T0=(D*E-B*F)/DET	06200000
	P0=(A*F-C*E)/DET	06210000
	IF (T0.LT.0.0.OR.P0.LT.0.0) GO TO 770	06220000
	IF (T0.GT.1.0.OR.P0.GT.1.0) GO TO 770	06230000
	XINT=T0*X2+(1.0-T0)*X1	06240000
	YINT=T0*Y2+(1.0-T0)*Y1	06250000
	J=J+1	06260000
	X(J)=XINT	06270000
	Y(J)=YINT	06280000
	INTSP(NINTS,2)=J	06290000
	I=I-1	06300000
	GO TO 510	06310000
780	IERR=3	06320000
	WRITE (6,540) IERR	06330000
	WRITE (6,550) SPMAP,SPPOLY	06340000
	WRITE (6,620) U1,V1,U2,V2	06350000
	GO TO 910	06360000
C		06370000
C		06380000
	790 CONTINUE	06390000
C		06400000
C		06410000
C	FIND MINIMUM X AND Y VALUES	06420000
	TXMIN=1.0E+9	06430000
	TYMIN=1.0E+9	06440000
	TXMAX=0.	06450000
	TYMAX=0.	06460000
	DO 800 I=1,J	06470000
	IF (X(I).LT.TXMIN) TXMIN=X(I)	06480000
	IF (Y(I).LT.TYMIN) TYMIN=Y(I)	06490000
	IF (X(I).GT.TXMAX) TXMAX=X(I)	06500000
	IF (Y(I).GT.TYMAX) TYMAX=Y(I)	06510000
800	CONTINUE	06520000
C	TRANSLATE COORDINATES	06530000
	DO 810 I=1,J	06540000
	X(I)=X(I)-TXMIN	06550000
	Y(I)=Y(I)-TYMIN	06560000
810	CONTINUE	06570000
	AREA=0.0	06580000
	NQPTS=0	06590000
	DO 830 I=1,NINTS	06600000
	IJ=INTSP(I,1)+1	06610000
	IK=INTSP(I,2)	06620000
	IF (IJ.GT.IK) GO TO 830	06630000
	NQPTS=NQPTS+1	06640000
	P(NQPTS)=X(IJ-1)+TXMIN	06650000
	Q(NQPTS)=Y(IJ-1)+TYMIN	06660000
	DO 820 II=IJ,IK	06670000
	NQPTS=NQPTS+1	06680000
	P(NQPTS)=X(II)+TXMIN	06690000



IF (DONUT) AREA=-AREA	07250000
NQ=MAX+1	07260000
WRITE (6,7004) LUCODE(1), LUCODE(3), SPCODE(1), SPCODE(3)	07270000
7004 FORMAT (' ALLOUT----- MAJOR ', A3, 2X, I4, 2X, ' MINOR ', A3, 2X, I4)	07280000
WRITE (14'IV4,850) MAPNO, MAPOL1(1), SPMP, SPCODE(1), SPCODE(3), AREA,	07290000
IXMIN, YMIN, XMAX, YMAX, MAX, (X(IJ), Y(IJ), IJ=1, MAX)	07300000
WRITE (6,870) SPMP, MAPOL1, LUAREA, SPCODE, FILL, AREA	07310000
WRITE(13'IV3,860) MAPNO, MAPOL1(1), SPMP, SPCODE(1), SPCODE(3),	073200
CSPPOLY(2), AREA	073201
WRITE(9,860) MAPNO, MAPOL1(1), SPMP, SPCODE(1), SPCODE(3), SPPOLY(2),	073202
*AREA	073203
IOVER=1	07330000
C	07340000
C CAN LOOP HERE ON MINOR POLYGONS	07350000
C	07360000
910 CONTINUE	07370000
920 WRITE (6,950) INSP, MAPOL1(1)	07380000
C REPLACEMENT IN CODE	07390000
FIND (12'1)	07400000
IV2=1	07410050
GO TO 80	07420000
930 CONTINUE	07430000
WRITE (6,950) INTZ	07440000
940 CONTINUE	07450000
950 FORMAT (' EOF ON FILE ', I5, ' MAJOR POLYGON= ', I5, '///')	07460000
WRITE (6,960)	07470000
960 FORMAT (' PROCESSING ENDED AT EOD')	07480000
C 4 EOF'S ON FILE 13	07490000
DO 980 J=1,4	07500000
WRITE (13'IV3,970) (ENDOF(I), I=1,6)	07510000
970 FORMAT (8A4,A2)	07520000
980 CONTINUE	07530000
C 4 EOF'S ON FILE 14	07540000
DO 1000 J=1,4	07550000
WRITE (14'IV4,990) (ENDOF(I), I=1,12)	07560000
990 FORMAT (20A4)	07570000
1000 CONTINUE	07580000
DO 1001 J=1,4	07590000
WRITE (15'IV5,971) (ENDOF(I), I=1,7)	07600000
971 FORMAT (7A4)	07610000
1001 CONTINUE	07620000
C IF (.NOT.PRTOUT) GOTO 1999	07630000
FIND (11'1)	07640010
IV1=1	07650015
WRITE (6,1109)	07660020
1109 FORMAT (' FILE-11---')	07670025
DO 1130 IJ=1,10	07680030
1110 READ (11'IV1,851) SPMP, SPCODE(2), SPCODE(1), SPCODE(3), SPPOLY(2), ARE	076900
1A, TXMIN, TYMIN, TXMAX, TYMAX, NQPTS, (P(I), Q(I), I=1, NQPTS)	07700040
851 FORMAT(2I5,A3,I4,I2,F15.2/4F9.0,I5/(8F9.0))	077001
FIND (11'IV1)	07710045
WRITE (6,1021) SPMP, SPCODE(2), SPCODE(1), SPCODE(3), SPPOLY(2), AREA,	077200
1A, TXMIN, TYMIN, TXMAX, TYMAX, NQPTS, (P(I), Q(I), I=1, NQPTS)	07730055
1130 CONTINUE	07740060
WRITE(6,1201)	077401
1201 FORMAT(' FILE-12---')	077402
IV2=1	077403
FIND (12'1)	077404
DO 1230 IJ=1,10	07750065
1210 READ (12'IV2,851) SPMP, SPCODE(2), SPCODE(1), SPCODE(3), SPPOLY(2), ARE	077800
1A, TXMIN, TYMIN, TXMAX, TYMAX, NQPTS, (P(I), Q(I), I=1, NQPTS)	07790085

	FIND (12*IV2)	07800090
	WRITE(6,1021) SPMAP,SPCODE(2),SPCODE(1),SPCODE(3),SPPOLY(2),AREA,	078100
	ITXMIN,TYMIN,TXMAX,TYMAX,NQPTS,(P(I),Q(I),I=1,NQPTS)	07820094
1230	CONTINUE	07830098
C	PRINT OUT 14 ON REQUEST	07840000
	FIND (14*1)	07850000
	IV4=1	07860050
	WRITE (6,1005)	07870060
1005	FORMAT (' FILE-14 ---')	07880070
	DO 1030 IJ=1,10	07890080
1010	READ (14*IV4,850)MAPNO,MAPOLI(1),SPMAP,SPCODE(1),SPCODE(3),AREA,	079000
	ITXMIN,TYMIN,TXMAX,TYMAX,NQPTS,(P(I),Q(I),I=1,NQPTS)	07910000
	FIND (14*IV4)	07920000
	WRITE (6,1020) MAPNO,MAPOLI(1),SPMAP,SPCODE(1),SPCODE(3),AREA,	079300
	ITXMIN,TYMIN,TXMAX,TYMAX,NQPTS,(P(I),Q(I),I=1,NQPTS)	07940000
1020	FORMAT (' ',3I5,A3,I4,F15.2/' ',4F10.0,I5/' ',8F10.0))	079500
1021	FORMAT (' ',2I5,A3,I4,I2,F15.2/' ',4F10.1,I5/' ',8F10.1))	079501
1030	CONTINUE	07960000
1999	CONTINUE	07970090
	STOP	07980000
C	END	07990000
C	POLYRD/TIDY	08000000
C	LATEST VERSION 9 JUNE '75	08010000
	SUBROUTINE POLYRD(*,MAPNO,POLYNO,CODE,AREA,XMIN,YMIN,XMAX,	08020000
1	YMAX,N,X,Y,INUNIT,FAC,DONUT,OUT,./INV/)	08030000
	DIMENSION X(1),Y(1)	08040000
	INTEGER MAPNO,POLYNO(2),CODE(3)	08050000
	LOGICAL DONUT,OUT	08060000
	LOGICAL IFLAG	08070000
	FAC=1.0	08080000
	DONUT=.FALSE.	08090000
10	CONTINUE	08100000
	INQ=INV	08110000
	READ (INUNIT*INV,20,ERR=150) MAPNO,CODE(2),CODE(1),CODE(3),ITYPE,	08120000
1	AREA,XMIN,YMIN,XMAX,YMAX,N,(X(K),Y(K),K=1,N)	08130000
20	FORMAT (2I5,A3,I4,I2,F15.2/4F9.0,I5/(8F9.0))	08140000
C	SOFTWARE EOF FLAG	081500
	IF (MAPNO.EQ.99999) GO TO 150	08160000
	IF (MAPNO.EQ.00000) GO TO 150	08170000
C	CALCULATE AREA FOR POLYGON	08180000
	IFLAG=.FALSE.	08190000
C	SKIP IF AREAOF CALLED PREVIOUSLY ON THIS P-GON	08200000
	IF (AREA)30,25,30	08210000
25	CALL AREAOF (AREA,N,X,Y,IFLAG)	08220000
	IF (.NOT. IFLAG)GOTO 30	08230000
	FIND (INUNIT*INQ)	08240000
	WRITE(6,200)MAPNO,CODE(2),CODE(3)	082500
200	FORMAT (' REVERSED POLYGON ',I5,2X,I5,2X,I4)	08260000
	WRITE (INUNIT*INQ,20) MAPNO,CODE(2),CODE(1),CODE(3),ITYPE	08270000
1	AREA,XMIN,YMIN,XMAX,YMAX,N,(X(K),Y(K),K=1,N)	08280000
30	CONTINUE	08290000
	IF (ITYPE.EQ.10) GO TO 40	08300000
	DONUT=.TRUE.	08310000
	GO TO 60	08320000
C	40 DO 50 J=2,N	08330000
C	IQ=J-1	08340000
C	X(IQ)=X(J)	08350000
C	Y(IQ)=Y(J)	08360000
C	50 CONTINUE	08370000
C	N=N-1	08380000

40	CONTINUE	08390000
	NOR=N	08400000
	ND2=N/2	08410000
60	POLYNO(1)=CODE(2)	08420000
	POLYNO(2)=ITYPE	08430000
	IF (X(1).NE.X(N)) GO TO 70	08440000
	IF (Y(1).EQ.Y(N)) GO TO 130	08450000
70	DIST=SQRT((ABS(X(1)-X(N)))**2.+(ABS(Y(1)-Y(N)))**2.)	08460000
	DIST2=SQRT((ABS(X(1)-X(N-1)))**2.+(ABS(Y(1)-Y(N-1)))**2.)	08470000
	IF (DIST.LE.DIST2) GO TO 110	08480000
	N=N-1	08490000
	IF (N.GT.ND2) GO TO 90	08500000
	N=NOR	08510000
	WRITE (6,80) N	08520000
80	FORMAT (' POLYGON PUT BACK IN ORIGINAL STATE N= ',I4)	08530000
	DIST=SQRT((ABS(X(1)-X(N)))**2.+(ABS(Y(1)-Y(N)))**2.)	08540000
	GO TO 110	08550000
90	WRITE (6,100) N	08560000
100	FORMAT (' POLYGON NOW HAS',I4,' PCINTS')	08570000
	GO TO 70	08580000
110	N=N+1	08590000
	X(N)=X(1)	08600000
	Y(N)=Y(1)	08610000
	IF (DIST.LT.10.) GO TO 130	08620000
	WRITE (6,120) DIST	08630000
120	FORMAT (' WARNING...CLOSING DISTANCE = ',F8.2)	08640000
130	CONTINUE	08650000
	IF (.NOT.DONUT) GO TO 140	08660000
	DONUT=.TRUE.	08670000
	OUT=.TRUE.	08680000
140	RETURN	08690000
150	RETURN 1	08700000
160	WRITE (6,165)INUNIT,INV	08710000
165	FORMAT (' AREA WRITE ERROR FROM POLYRD',/, ' FILE= ',I5,3X,	08720000
	1 ' RECORD= ',I5,'--RUN TERMINATED--')	08730000
	STOP	08740000
C		08750000
	END	08760000
C		08770000
C	SUBROUTINE AREAOF	08780000
C	LATEST VERSION 9 JUNE '75	08790000
C	CALCULATES AREA OF ANY POLYGON USING VECTOR ANALYSIS METHOD.	08800000
C	AREA= AREA TO BE DETERMINED	08810000
C	AND DIRECTION OF POINTS IE CLOCKWISE OR COUNTER-CLOCKWISE	08820000
C	N= NUMBER OF POINTS	08830000
C	X= ARRAY OF X VALUES	08840000
C	Y= ARRAY OF Y VALUES	08850000
C	IFLAG= DIRECTION FLAG	08860000
C		08870000
	SUBROUTINE AREAOF(AREA,N,X,Y,IFLAG)	08880000
	DIMENSION X(1),Y(1),X1(1501),Y1(1501)	08890000
	LOGICAL IFLAG	08900000
C	INDICATES POLYGON WAS CLOCKWISE OR NOT	08910000
	IFLAG=.FALSE.	08920000
	AREA=0.0	08930000
	M=N-1	08940000
	DO 50 L1=2,M	08950000
	A=X(L1)-X(1)	08960000
	B=Y(L1)-Y(1)	08970000
	A1=X(L1+1)-X(1)	08980000
	B1=Y(L1+1)-Y(1)	08990000

[illegible]



C		09610000
	10 CONTINUE	09620000
C		09630000
	IF (DIST .LE. 20000.) GO TO 20	09640000
C		09650000
	I = I + 1	09660000
C		09670000
	HALVE THE MAGNITUDE OF THE VECTOR	09680000
C		09690000
	UNTIL THE MAGNITUDE IS LESS THAN	09700000
C		09710000
	OR EQUAL TO THE VALUE OF .1	09720000
C		09730000
	NOTE... DIST IS THE SQUARE OF THE	09740000
C		09750000
	MAGNITUDE.	09760000
	DIST = DIST / 4.	09770000
C		09780000
	GO TO 10	09790000
C		09800000
	20 CONTINUE	09810000
C		09820000
	COMPUTE THE SLOPE, IF DEFINED	09830000
C		09840000
	IF(ABS(DLX) .LT..005 ) GO TO 30	09850000
C		09860000
	SLOPE = DLY / DLX	09870000
C		09880000
	COMPUTE INCREMENTAL VECTORS	09890000
C		09900000
	SPX = SIGN (SQRT(DIST / (1. + SLOPE ** 2)), DLX)	09910000
C		09920000
	SPY = SLOPE * SPX	09930000
C		09940000
	GO TO 40	09950000
C		09960000
	30 CONTINUE	09970000
C		09980000
	COMPUTE THE SPY VECTOR INCREMENT	10000000
C		10010000
	IF SLOPE IS UNDEFINED	10020000
	SPX = 0.	10030000
C		10040000
	SPY = SIGN (SQRT(DIST), DLY)	10050000
C		10060000
	40 CONTINUE	10070000
C		10080000
	NUMBER OF INCREMENTS	10090000
C		10100000
	NOSTP = 2 ** I	10110000
C		10120000
	50 CONTINUE	10130000
	RETURN	10140000
C		10150000
	END	10160000
	FUNCTION LEVEL(Y,YMIN,STEP,ISTP,YMAX)	10170000
C		10180000
	BLOW=YMIN	10190000
	DO 10 I=2,ISTP	10200000
C		
	BLOW=BLOW+STEP	
C		
	BHIH=BLOW+STEP	
	IF(I .EQ. ISTP) BHIH = YMAX	
C		
	IF(Y.LE.BLOW) GO TO 10	
	IF(Y.GT.BHIH) GO TO 10	
C		
	LEVEL=I	
	GO TO 20	
C		
	10 CONTINUE	
	LEVEL=1	
C		
	20 RETURN	
	END	
C	!!!!!!	
	SUBROUTINE PIP(PP,QQ,IND,X,Y,N)	
C		
	DIMENSION X(N),Y(N)	
	LOGICAL SWT	
C		
	*****	
C		

```

C
C
C * * * * * 10220000
C POINT-IN-POLYGON SUBROUTINE 10230000
C * * * * * 10240000
C 10250000
C 10260000
C 10270000
C 10280000
C THE PURPOSE OF THIS SUBROUTINE IS TO DETERMINE IF THE POINT, 10290000
C (PP,QQ), IS IN THE POLYGON DEFINED BY THE ARRAYS X AND Y, 10300000
C WHICH ARE OF LENGTH N. IND=0 IF THE POINT IS NOT IN THE 10310000
C POLYGON, AND IND=1 IF THE POINT IS IN THE POLYGON 10320000
C 10330000
C THE METHOD USED IS TO DRAW A DIRECTED LINE SEGMENT FROM THE 10340000
C POINT (PP,QQ) TO (INFINITY,QQ). HENCE THE LINE STARTS FROM 10350000
C (PP,QQ) AND LIES IN THE PLUS X DIRECTION WITH SLOPE ZERO. 10360000
C IF THIS DIRECTED LINE SEGMENT CROSSES THE POLYGON SIDES AN ODD 10370000
C NUMBER OF TIMES THE POINT (PP,QQ) IS IN THE POLYGON. IF THE 10380000
C DIRECTED LINE SEGMENT CROSSES THE POLYGON SIDES AN EVEN NUMBER OF 10390000
C TIMES THE POINT LIES OUTSIDE OF THE POLYGON. IF THE POINT 10400000
C LIES ON THE SIDE OF THE POLYGON IT IS CONSIDERED IN THE POLYGON. 10410000
C 10420000
C SWT IS SET TO FALSE. AT THE 10430000
C END OF THE ROUTINE IF SWT IS TRUE 10440000
C THE POINT IS CONSIDERED IN THE 10450000
C POLYGON. EACH TIME IT IS 10460000
C DETERMINED THAT THE DIRECTED LINE 10470000
C SEGMENT CROSSES THE POLYGON SIDE 10480000
C THE VALUE OF SWT IS FLIPPED. HENCE 10490000
C IF THE NUMBER OF CROSSINGS IS ODD 10500000
C THE VALUE OF SWT WILL BE TRUE. 10510000
C 10520000
C SWT=.FALSE. 10530000
C 10540000
C IN CONSIDERING THE POINTS IN 10550000
C THE POLYGON ARRAYS, POINTS I AND 10560000
C I+1 ARE CONSIDERED IN EACH LOOP. 10570000
C THEREFORE THE LOOP LIMIT 10580000
C PARAMETER IS SET TO N-1 10590000
C 10600000
C M=N-1 10610000
C 10620000
C AT TIMES IN THE ALGORITHM IT IS 10630000
C NECESSARY TO KEEP TRACK OF THE LAST 10640000
C Y VERTEX COORDINATE. SINCE ALL Y 10650000
C VALUES ARE POSITIVE -1 IS USED AS A 10660000
C NULL VALUE 10670000
C 10680000
C YREM=-1 10690000
C 10700000
C 10710000
C IND IS SET TO ZERO AND CHANGED 10720000
C TO ONE LATER IF NEED BE 10730000
C 10740000
C IND=0 10750000
C 10760000
C 10770000
C LOOP THRU THE POLYGON POINTS 10780000
C 10790000
C 10800000
C DO 10 I=1,M 10810000
C 10820000
C 10830000
C STORE ARRAY VARIABLES INTO 10840000
C SCALARS TO ELIMINATE UNNECESSARY 10850000
C INDEXING 10860000
C 10870000
C YP=Y(I) 10880000
C YYP=Y(I+1) 10890000
C XP=X(I) 10900000
C XXP=X(I+1) 10910000
C 10920000
C 10930000
C THE LINE SEGMENT DEFINED BY (XP,YP) AND (XXP,YY) IS CONSIDERED 10940000

```

C		10830000
C		10840000
C	IF THE POINT IS COMPLETELY ABOVE	10850000
C	THE LINE SEGMENT THERE WILL	10860000
C	BE NO CROSSING.	10870000
C	IF(QQ.GT.YP.AND.QQ.GT.YYP) GO TO 77	10880000
C		10890000
C	LIKEWISE IF THE POINT IS BELOW	10900000
C		10910000
C	IF(QQ.LT.YP.AND.QQ.LT.YYP) GO TO 77	10920000
C		10930000
C	OR COMPLETELY TO THE RIGHT OF BOTH	10940000
C	POINTS XP AND XYP	10950000
C		10960000
C	IF(PP.GT.XP.AND.PP.GT.XYP) GO TO 77	10970000
C		10980000
C		10990000
C	*****	11000000
C	CONSIDER THE SPECIAL CASE WHEN QQ=YP OR BRANCH AROUND.	11010000
C	IN THIS CASE THE POLYGON SIDE CAN COME UP TO THE POINT.	11020000
C	TURN AROUND AND GO BACK, OR IT CAN ACTUALLY CROSS IT	11030000
C		11040000
C	*****	11050000
C	IF(QQ.NE.YP) GO TO 20	11060000
C		11070000
C	ARE THE POINTS (PP,QQ) AND (XP,YP)	11080000
C	THE SAME. IF SO THE POINT IS	11090000
C	CONSIDERED IN.	11100000
C	IF(PP.EQ.XP) GO TO 30	11110000
C		11120000
C	IF THIS VERTEX WAS INCOUNTED LAST	11130000
C	TIME THRU THE LOOP THEN BRANCH OUT;	11140000
C	9999 CONTINUE	11150000
C		11160000
C	IF(QQ.EQ.YREM) GO TO 40	11170000
C		11180000
C	REMEMBER THE VERTEX SO IT CAN BE	11190000
C	SKIPPED NEXT TIME THRU.	11200000
C	YREM=YP	11210000
C		11220000
C	SEARCH BACKWARD UNTIL THERE	11230000
C	IS A VERTEX POINT WITH A Y VALUE	11240000
C	NOT QQ	11250000
C	LAST=I	11260000
C	50 LAST=LAST-1	11270000
C	IF(LAST.EQ.0)GO TO 10	11280000
C	IF(QQ.EQ.Y(LAST)) GO TO 50	11290000
C	YLAST=Y(LAST)	11300000
C		11310000
C	SEARCH FOWARD UNTIL THERE IS A	11320000
C	VERTEX POINT WITH A Y VALUE NOT QQ.	11330000
C	NEXT=I	11340000
C	60 NEXT=NEXT+1	11350000
C	IF(NEXT.GT.N)GO TO 10	11360000
C	IF(QQ.EQ.Y(NEXT)) GO TO 60	11370000
C	YNEXT=Y(NEXT)	11380000
C		11390000
C	IF QQ IS GREATER THAN BOTH THEN	11400000
C	THERE IS NO CROSSING.	11410000
C		11420000
C	IF(QQ.GT.YLAST.AND.QQ.GT.YNEXT) GO TO 10	11430000
C		
C	IF QQ IS LESS THAN BOTH THERE	
C	IS NO CROSSING	
C		
C	IF(QQ.LT.YLAST.AND.QQ.LT.YNEXT) GO TO 10	

C		11440000
C	CHECK FOR CROSSING	11450000
	IF(PP.LT.XP) GO TO 70	11460000
	GO TO 10	11470000
C		11480000
C		11490000
C	*****	11500000
C	CONSIDER THE SPECIAL CASE WHEN QQ=YYP OR BRANCH AROUND.	11510000
C	IN THIS CASE THE POLYGON SIDE CAN COME UP TO THE POINT,	11520000
C	TURN AROUND AND GO BACK, OR IT CAN ACTUALLY CROSS IT	11530000
C	*****	11540000
C		11550000
	20 IF(QQ.NE.YYP) GO TO 80	11560000
C		11570000
C	ARE THE POINTS(PP,QQ) AND (XXP,YYP)	11580000
C	THE SAME. IF SO THE POINT IS	11590000
	CONSIDERED IN.	11600000
	IF(PP.EQ.XXP) GO TO 30	11610000
C		11620000
C	IF THIS VERTEX WAS INCOUNTED LAST	11630000
C	TIME THRU THE LOOP THEN BRANCH OUT;	11640000
C	OTHERWISE THE CROSSING GETS COUNTED	11650000
	TWICE.	11660000
	IF(QQ.EQ.YREM) GO TO 40	11670000
C		11680000
C	REMEMBER THE VERTEX SO IT CAN BE	11690000
C	SKIPPED NEXT TIME THRU.	11700000
	YREM=YYP	11710000
C		11720000
C	SEARCH BACKWARD UNTIL THERE	11730000
C	IS A VERTEX POINT WITH A Y VALUE	11740000
	NOT QQ	11750000
	LAST=I+1	11760000
	90 LAST=LAST-1	11770000
	IF(QQ.EQ.Y(LAST)) GO TO 90	11780000
	YLAST=Y(LAST)	11790000
C		11800000
C	SEARCH FOWARD UNTIL THERE IS A	11810000
	VERTEX POINT WITH A Y VALUE NOT QQ.	11820000
	NEXT=I+1	11830000
	100 NEXT=NEXT+1	11840000
	IF(QQ.EQ.Y(NEXT)) GO TO 100	11850000
	YNEXT=Y(NEXT)	11860000
C		11870000
C	IF QQ IS GREATER THAN BOTH THEN	11880000
C	THERE IS NC CROSSING.	11890000
	IF(QQ.GT.YLAST.AND.QQ.GT.YNEXT) GO TO 10	11900000
C		11910000
C	IF QQ IS LESS THAN BOTH THERE	11920000
C	IS NO CROSSING	11930000
	IF(QQ.LT.YLAST.AND.QQ.LT.YNEXT) GO TO 10	11940000
C		11950000
C	CHECK FOR CROSSING	11960000
	IF(PP.LT.XXP) GO TO 70	11970000
	GO TO 10	11980000
C		11990000
	40 IF(YP.NE.YYP) YREM=-1.0	12000000
	GO TO 10	12010000
C		12020000
C	IF THE PROGRAM FLOW GETS TO HERE	12030000
C	AND PP IS COMPLETELY TO THE LEFT OF	12040000
C	THE LINE SEGMENT THERE IS A	
C	CROSSING	
	80 YREM=-1.0	
	IF(PP.LE.XP.AND.PP.LE.XXP) GO TO 70	

C		12050000
C		12060000
C		12070000
C		12080000
C	IF(XP.EQ.XXP) GO TO 30	12090000
C		12100000
C		12110000
C		12120000
C	IF(YP.EQ.YYP) GO TO 30	12130000
C		12140000
C		12150000
C		12160000
C		12170000
C		12180000
C		12190000
C	XTEMP=(XP-XXP)*(QQ-YYP)/(YP-YYP)+XXP	12200000
C		12210000
C		12220000
C		12230000
C	IF(PP.GT.XTEMP) GO TO 10	12240000
C		12250000
C		12260000
C	IF(PP.EQ.XTEMP) GO TO 30	12270000
C		12280000
C		12290000
C	70 SWT=.NOT.SWT	12300000
C	GO TO 10	12310000
C		12320000
C		12330000
C	30 SWT=.TRUE.	12340000
C	GO TO 200	12350000
C	77 YREM=-1.0	12360000
C	10 CONTINUE	12370000
C	200 CONTINUE	12380000
C		12390000
C		12400000
C		12410000
C		12420000
C		12430000
C		12440000
C		12450000
C		12460000
C	CLUGE	12470000
C	ROUTINE TO READ CARDSFOR MAJOR,MINOR POLYGONS ONTO FILE11,	12480000
C	AND FILE12	12490000
C	SUBROUTINE CLUGE	12500000
C	INTEGER IBUF(20)	12510000
C	INTEGER MINR/'MINR'//,IEND/'END '//,MORE/'MORE'//	125101
C	INTEGER NINE,NIN,BLANK(17)	125102
C	DATA NINE /'9999'//, NIN /'9 '//, BLANK /17*' '//	125103
C	DATA ICT /'1378'//	12520000
C	COMMON /ALPHA/IV1,IV2,IV3,IV4,IV5	12530000
C	IV1=1	12540000
C	IV2=1	12550000
C	FIND (11'1)	12560000
C	FIND (12'1)	12570000
C		12580000
C	START BY READING MAJOR POLY CARDS	125900
C	MAJOR FIRST	125901
C	10 READ (3,100,END=20) (IBUF(I),I=1,20)	12610000
C	IF (IBUF(1).LT.ICT) GO TO 10	12680000
C	100 FORMAT (20A4)	
C	WRITE (6,111)(IBUF(I),I=1,20),IV1	

WRITE (11*IV1,100)(IBUF(I),I=1,20)	12690000
111 FORMAT (' FROM 11 ',20A4,' IV1= ',I6)	12700000
FIND (11*IV1)	12710000
GO TO 10	12720000
20 CONTINUE	127300
DO 21 J=1,6	127301
21 WRITE(11*IV1,100) NINE,NIN,BLANK	127302
30 READ (4,100,END=40) (IBUF(I),I=1,20)	127303
WRITE(12*IV2,100) (IBUF(I),I=1,20)	127304
C WRITE (6,112)(IBUF(I),I=1,20)	12740000
112 FORMAT (' FROM 12 ',20A4,' IV2= ',I6)	127500
FIND (12*IV2)	12760000
GO TO 30	127700
40 CONTINUE	12830000
DO 31 J=1,5	128301
31 WRITE(12*IV2,100) NINE,NIN,BLANK	128302
FIND (11*1)	12840000
FIND (12*1)	12850000
IV1=1	12860000
IV2=1	12870000
DO 511 IJ=1,10	12880000
READ (11*IV1,100)(IBUF(I),I=1,20)	12890000
FIND (11*IV1)	12900000
WRITE (6,111) (IBUF(I),I=1,20),IV1	129100
511 CONTINUE	12920000
DO 611 IJ=1,10	12930000
READ (12*IV2,100)(IBUF(I),I=1,20)	12940000
FIND (12*IV2)	12950000
WRITE (6,112) (IBUF(I),I=1,20),IV2	129600
611 CONTINUE	12970000
RETURN	12980000
END	12990000

THIS PAGE IS  
OF POOR QUALITY

APPENDIX VIII  
MAJOR POLYGON DATA REPORT PROGRAM LISTING

```

DIMENSION PTAREA(30), PERCT(30), PDIFF (30)
INTEGER SUMAP(30),SUCODE(30),SUORDR(30),SUTYPE(30)
INTEGER CTRNO,BLKNO,CTORDR,CTTYPE
INTEGER BLANK
LOGICAL SKIP
DATA BLANK /4H
SKIP = .FALSE.
LINECT=0
1 CONTINUE
WRITE (6,2001)
2001 FORMAT(1H1,52X, 'S U M M A R Y O U T P U T'/
*48X, 'LOS ANGELES-SANTA MONICA STUDY AREA')
WRITE (6,2002)
2002 FORMAT(56X, 'MAJOR POLYGON FILE')
WRITE (6,2003)
2003 FORMAT(1X, 'CENSUS CENSUS POLY POLY LAND USE LAND USE '
1,'POLY POLY ORIGINAL AREA RESIDUAL AREA DIFFERENCE',
2,' PERCENT NUM OF')
WRITE (6,2004)
2004 FORMAT(1X, 'TRACT BLOCK NUM TYPE MAP CODE '
2,'NUM TYPE',61X,'RES POLY')
WRITE (6,2005)
2005 FORMAT (1H+ '-----'
2,'-----'
3,'-----' '/')
IF(LINECT.EQ.0)GO TO 5
LINECT=0
GO TO 59
5 INUM = 0
PTAREA (1) = 0.0
PERCT(1) = 0.0
PDIFF(1)=0.0
SUMAP(1)=0
SUCODE(1)=0
SUORDR(1)=0
SUTYPE(1)=0
C READ (15*IV5,101) CTRNO,BLKNC,CTORDR,CTTYPE,CTAREA
READ (15 ,101) CTRNO,BLKNC,CTORDR,CTTYPE,CTAREA
101 FORMAT (2I5,I4,I2,F15.2)
IF (CTRNO.EQ.99999) GO TO 999
TOAREA = TOAREA + CTAREA
IF(SKIP) GOTO 15
C 10 READ(13*IV3,102) MAPNO,MAPNO1,LUMAP,LUCODE,LUORDR,LUTYPE,AREA
10 READ(13 ,102) MAPNO,MAPNO1,LUMAP,LUCODE,LUORDR,LUTYPE,AREA
102 FORMAT(3I5,A3,I4,I2,F15.2)
15 CONTINUE
SKIP = .FALSE.
IF (MAPNO-CTRNC) 10,20,30
20 IF(MAPNO1-BLKNC) 10,40,30
40 INUM = INUM + 1
PTAREA(INUM) = AREA
PERCT(INUM) = 100.*AREA/CTAREA
PDIFF(INUM)=CTAREA-AREA
SUMAP(INUM)=LUMAP
SUCODE(INUM)=LUCODE
SUORDR(INUM)=LUORDR
SUORDR(INUM)=LUORDR
SUTYPE(INUM)=LLTYPE
IF(PERCT(INUM).LT.0.009) PERCT(INUM)=0.00
IF (LUTYPE.LE.10) GO TO 50
PTAREA(INUM) = -PTAREA(INUM)

```



```

PERCT (INUM) = -PERCT (INUM)
PDIFF(INUM) = -PDIFF(INUM)
50 CONTINUE
TOTPR = TOTPR + PTAREA(INUM)
GO TO 10
30 CONTINUE
MCTRNO = MOD (CTRNO,10)
MBLKNO = MOD (BLKNO,10)
CTRNO = CTRNO/10*100+MCTRNO
BLKNO = BLKNO/10*100+MBLKNO
STOT=0.0
SDIFF=0.0
SPERD=0.0
IF(LINECT.GT.50)GO TO 1
59 CONTINUE
IF(INUM.LE.0) GO TO 61
DO 60 J=1,INUM
LINECT=LINECT+1
65 CONTINUE
WRITE(6,201) SUMAP(J),SUCODE(J),
*SUORCR(J),SUTYPE(J),PTAREA(J)
201 FORMAT(34X, 15.7X,A3,6X,I2,4X,I9,18X,F18.2)
STOT=STOT+PTAREA(J)
60 CONTINUE
SDIFF=CTAREA-STOT
SPERD=100.*SDIFF/CTAREA
WRITE(6,203) CTRNO,BLKNO,CTORCR,CTTYPE,CTAREA,STOT,SDIFF,SPERD,
*INUM
203 FORMAT(1X,I6,4X,I5,3X,I2,I6,6X, 'TOTAL'
*,2F18.2,F13.2,F9.2,I8/)
GO TO 62
61 CONTINUE
WRITE(6,204) CTRNO,BLKNO,CTORCR,CTTYPE,CTAREA
204 FORMAT(1X,I6,4X,I5,3X,I2,4X,I2,36X, 1F18.2,
*4X,*** NO RESIDUAL OVERLAY POLYGON FOUND ***/)
62 CONTINUE
LINECT=LINECT+2
SKIP = .TRUE.
GO TO 5
999 CONTINUE
WRITE(6,202) TCAREA,TOTPR
202 FORMAT(1H0,'TOTAL AREA OF CENSUS TRACTS=',F15.2/
* 1H0,'TOTAL AREA OF RESIDUAL POLYGONS=',F15.2)
END

```

PAGE IS  
 QUALITY

APPENDIX IX  
MINOR POLYGON DATA REPORT PROGRAM LISTING

```

CFILE 10=TD/SORTED/RESID13,UNIT=DISK,RECORD=15,BLOCKING=40
CFILE 11=TD/LU13E,UNIT=DISK,RECORD=15,BLOCKING=40
C   DEFINE FILE 10 (3000,80,E,I1)
C   DEFINE FILE 11 (3000,80,E,I2)
    DIMENSION SAREA(30),ISBLK(30)
    I1=1
    I2=1
    IERRS=0
    ICTR=0
    JCTR=0
    JUMP=0
    ASSIGN 100 TO JUMP
    ASSIGN 200 TO IGO
    AREA1=0.
    AREA2=0.
    TOT1=0.
    TOT2=0.
    IITA2=0
    ILINES=0
1  WRITE (6,2021)
2021 FORMAT(1H1,52X, 'SUMMARY OUTPUT'/
*48X, 'LOS ANGELES-SANTA MONICA STUDY AREA')
    WRITE (6,2022)
2022 FORMAT(56X, 'MINOR POLYGON FILE')
    WRITE (6,2023)
2023 FORMAT(1X,'CENSUS    CENSUS          LAND USE    LAND USE POLY    POLY'
1,'      ORIGINAL AREA    RESIDUAL AREA    DIFFERENCE ')
2 '      PERCENT    NUM OF')
    WRITE (6,2024)
2024 FORMAT(1X, 'TRACT    BLOCK          MAP          CODE    NUM    '
2'TYPE',70X,'RES POLY')
    WRITE (6,2025)
2025 FORMAT (1H+ '-----'
1,'--'
2,'-----')
    WRITE(6,2001)
    IF(ILINES.EQ.0)GO TO 5
    GO TO 205
C   READ(10'I1,1000) IMAP,ITAZ,ILUC,IPOLY,ITYPE,AREA1
    5 READ(10 ,1000) IMAP,IBLK,ITAZ,ILUC,IPOLY,ITYPE,AREA1
1000 FORMAT (15,15,15,A3,14,12,   F15.2)
    ILINES=ILINES+6
    INUM=1
C   FIND(10'I1)
    ICTR=ICTR+1
C   33 READ(11'I2,1000) JMAP,JTAZ,JLUC,JPOLY,JTYPE,AREA2,N
    READ(11 ,1001) JMAP,JTAZ,JLUC,JPOLY,JTYPE,AREA2,N
1001 FORMAT (15,   15,A3,14,12,   F15.2/ 36X,15)
C   FIND(11'I2)
    A=N
    A=(A/5.0)+.9
    N=A
    I1=I1+N+1
    JCTR=JCTR+1
C   10 READ(10'I1,1000) IIMAP,IITAZ,IILUC,IIPOLY,IITYPE,AREA1,N
    10 READ(10 ,1000) IIMAP,IIBLK,IITAZ,IILUC,IIPOLY,IITYPE,AREA1
    IF(IIMAP.EQ.99999) GO TO 900
C   FIND(10'I1)
    ICTR=ICTR+1
    20 IF(IIMAP.NE.IMAP)GO TO 30
    IF(IILUC.NE.ILUC)GO TO 30

```

```

IF(IIPOLY.NE.IPOLY)GO TO 30
IF(IITYPE.NE.ITYPE)GO TO 30
ISBLK(1)=IBLK
SAREA(1)=AREA1
AREA1=AREA1+AREA1
INUM=INUM+1
SAREA(INUM)=AREA1
ISBLK(INUM)=IBLK
GO TO 10
30 GO TO JUMP,(100,140,200)
40 IMAP=IMAP
IBLK=IBLK
ITAZ=ITAZ
ILUC=ILUC
IPOLY=IPOLY
AREA1=AREA1
ITYPE=ITYPE
INUM=1
GO TO 10
C 100 READ(11*I2,1000)JJMAP,JJTAZ,JJLUC,JJPOLY,JJTYPE,AREAJ,N
100 READ(11,1001)JJMAP,JJTAZ,JJLUC,JJPOLY,JJTYPE,AREAJ,N
IF(JJMAP.EQ.99999) GO TO 900
A=N
A=(A/5.0)+.9
N=A
I2=I2+N+1
C FIND(11*I2)
JCTR=JCTR+1
IF(JJTYPE.GT.10)AREAJ=-AREAJ
120 IF(JJMAP.NE.JMAP)GO TO 130
IF(JJLUC.NE.JLUC)GO TO 130
IF(JJPOLY.NE.JPOLY)GO TO 130
IF(JJTYPE.NE.JTYPE)GO TO 130
AREA2=AREA2+AREAJ
GO TO 100
130 GO TO 1GO,(200)
140 JTYPE=JJTYPE
JMAP=JJMAP
JTAZ=JJTAZ
JLUC=JJLUC
JPOLY=JJPOLY
AREA2=AREAJ
GO TO 100
200 IF(IPOLY.NE.JPOLY)GO TO 320
ASSIGN 140 TO JUMP
DELTA=AREA2-AEA1
IF(AREA2.EQ.0.0)GO TO 210
DPCT=(ABS(DELTA)/AREA2)*100.0
IF(ILINES.LT.50)GO TO 210
GO TO 1
205 ILINES=5
210 CONTINUE
DO 215 J=1,INUM
MBLKNO = MOD (ISBLK(J),10)
ISBLK(J) = ISBLK(J)/10*100+MBLKNO
WRITE(6,2015) ISBLK(J),JMAP,JLUC,JPOLY,JTYPE,SAREA(J)
2015 FORMAT(11X,15,8X,16,7X,A3,6X,I2,4X,I2,8X,F13,2)
ILINES=ILINES+1
215 CONTINUE
MIMAP = MOD (IMAP,10)
IMAP = IMAP/10*100+MIMAP

```

THIS PAGE IS  
 OF POOR QUALITY

```

WRITE(6,2002)IMAP,JMAP,JLUC,JPOLY,JTYPE,AREA2,AREA1,DELTA,
DPCT,INUM
ILINES=ILINES+1
TOT1=TOT1+AREA1
TOT2=TOT2+AREA2
IF(DELTA.NE.0.0)IERRS=IERRS+1
GO TO 40
320 IF(IPOLY.LT.JPOLY)GO TO 350
GO TO 360
350 ASSIGN 200 TO JUMP
WRITE(6,2001)
WRITE(6,2003)IMAP,      ILUC,IPOLY,ITYPE,AREA1
WRITE(6,2001)
GO TO 40
360 WRITE(6,2001)
WRITE(6,2004)JMAP,      JLUC,JPOLY,JTYPE,AREA2
WRITE(6,2001)
GO TO 140
900 WRITE(6,2001)
DELTA=TOT2-TOT1
DPCI=(ABS(DELTA)/TOT2)*100.0
WRITE(6,2005)TOT2,TOT1,DELTA,DPCCT
A=IERRS
B=JCTR
C=(A/B)*100.0
WRITE(6,2006)JCTR,ICTR,IERRS,C
STOP
2000 FORMAT('1','MAP      TAZ      LUC  POLYGON  TYPE  ORIGINAL AREA',
1      ' RESIDUAL AREA  DIFFERENCE  PERCENT',
2      ' RESIDUAL POLYGONS')
2001 FORMAT(1X)
2002 FORMAT(I7,18X,I5,7X,A3,6X,I2,4X,I2,6X,F15.2,5X,F15.2,5X,
1F10.2,5X,F6.2,7X,I2/)
2003 FORMAT(25X,I5,7X,A3,6X,I2,4X,I2,6X,F15.2,4X,
1      '**** UNMATCHED RESIDUAL POLYGON ****')
2004 FORMAT(25X,I5,7X,A3,6X,I2,4X,I2,6X,F15.2,4X,
1      '**** UNMATCHED ORIGINAL POLYGON ****')
2005 FORMAT('0','**** OVERALL TOTALS ****',10X,F10.2,6X,F10.2,6X,
1      F10.2,4X,F6.2)
2006 FORMAT(1X,'ORIGINAL POLYGONS',I5,' RESIDUAL POLYGONS',I5,
1      ' ERRORS',I5,' PERCENT',F7.2)
END

```